WM_W800_寄存器手册

V2.1

北京联盛德微电子有限责任公司（winner micro）

地址：北京市海淀区阜成路 67 号银都大厦 18 层

电话：+86-10-62161900

公司网址：www.winnermicro.com

文档修改记录

版本	修订时间	修订记录	作者	审核
V0.1	2019-09-25	创建	chengsheng	
V0.2	2020-05-13	修订遗留描述错误；	Chengsheng	
V1.0	2020-7-20	更新数字功能接口	Ray	
V2.0	2020-8-20	完善高速接口功能	Ray	
V2.1	2020-09-08	增加 SD_ADC 模块描述	chengsheng	

目录

图目录

表目录

1 引言

1.1 编写目的

W800 芯片是联盛德微电子推出的一款嵌入式 Wi-Fi SoC 芯片。该芯片集成度高，所需外围器件少，性价比高。适用于 IoT（智能家庭）领域各种智能产品。高度集成的 Wi-Fi 和蓝牙 4.2 Combo 功能是其主要功能；另外，该芯片集成 XT804 内核，内置 QFlash，SDIO、SPI、UART、GPIO、I²C、PWM、I²S、7816，LCD，Touch Sensor 等接口，支持多种硬件加解密算法。此外，该芯片 MCU 含有安全内核，支持代码安全权限设置，全系统支持固件加密存储，固件签名，安全调试，安全升级等各种安全措施，提升产品安全特性。

本文档主要描述 W800 芯片内部结构，各功能模块信息以及详细寄存器使用信息；是开发者开发驱动程序、应用程序的主要参考资料。联盛德微电子提供的 SDK 中有各种功能的开源实现，开发者可以参考对应驱动程序、应用程序示例，以增加对芯片功能和寄存器描述的理解。本文档没有对 Wi-Fi/BT 部分的寄存器描述。

1.2 参考资料

W800 芯片封装参数、电气特性、射频参数等信息，请参考《W800 芯片产品规格书》；

W800 芯片集成了 ROM 程序，ROM 程序提供了下载固件，MAC 地址读写、Wi-Fi 参数读写等功能，详细信息请参考《WM_W800_ROM 功能简述》；

W800 芯片内置了 2Mbytes QFlash 存储器，作为代码以及参数的存储空间。本文档提供了 QFlash 基本操作信息。如有超出本文档范围的需求，需要参考 QFlash 手册；

W800 芯片采用杭州平头哥 XT804 核心，804 相关的功能介绍、开发资料等可以参考平头哥公司发布信息；

更多信息请参考联盛德微电子网站(http://www.winnermicro.com/)。

2 特征

- 芯片封装
  - QFN32 封装，4mm x 4mm。

- 芯片集成度
  - 集成 XT804 处理器，最高 240MHz
  - 集成 288KB SRAM
  - 集成 2MB FLASH
  - 集成 8 通道 DMA 控制器，支持 16 个硬件申请，支持软件链表管理
  - 集成 PA/LNA/TR-Switch
  - 集成 32.768KHz 时钟振荡器
  - 集成电压检测电路
  - 集成 LDO
  - 集成上电复位电路

- 芯片接口
  - 集成 1 个 SDIO2.0 Device 控制器，支持 SDIO 1 位/4 位/SPI 三种操作模式，工作时钟范围 0~50MHz

- 集成 1 个 SDIO 2.0 HOST 控制器，支持 SDIO 与 SD 卡操作，工作时钟范围 0~50MHz
- 集成 1 个 QSPI PSRAM 接口，支持最大 64MB 容量 PSRAM，最高工作时钟频率 80MHz；
- 集成 5 个 UART 接口，支持 RTS/CTS，波特率范围 1200bps~2Mbps
- 集成 1 个高速 SPI 从设备接口，工作时钟范围 0~50MHz
- 集成 1 个 SPI 主/从接口，主设备工作时钟最高为 20MHz，从设备支持最高 6Mbps 数据传输速率
- 集成一个 I2C 控制器，支持 100/400Kbps 速率
- 集成 PWM 控制器，支持 5 路 PWM

  单独输出或者 2 路 PWM 输入。最高输出频率 20MHz，最高输入频率 20MHz
- 集成双工 I2S 控制器，支持 32KHz 到 192KHz I2S 接口编解码
- 集成 1 个 7816 接口，兼容 UART 接口支持 ISO-7816-3 T=0.T=1 模式；支持

  EVM2000 协议
- 支持多种硬件加解密模式，包括

  RSA/AES/RC4/DES/3DES/RC4/SHA1/MD5/CRC8/CRC16/CRC32/TRNG
- 集成 1 路差分，或者两路单端 16bit ADC 接口；
- 集成 11 路 Touch Sensor；
- 支持最多 17 个 GPIO 口，每个 IO 口都有

  丰富的复用关系。具备输入输出配置选项。

- WIFI 协议与功能
  - 支持 GB15629.11-2006、IEEE802.11 b/g/n；
  - 支持 WMM/WMM-PS/WPA/WPA2/WPS
  - 支持 WiFi Direct；

- 支持 EDCA 信道接入方式；
- 支持 20/40M 带宽工作模式；
- 支持 STBC、GreenField、Short-GI、支持反向传输；
- 支持 RIFS 帧间隔；
- 支持 AMPDU、AMSDU；
- 支持 802.11n MCS 0~7、MCS32 物理层传输速率档位，传输速率最高到 150Mbps；
- 支持 HT-immediate Compressed BlockAck、nornal ACK、no ACK 应答方式；
- 支持 CTS to self；
- 支持 AP 功能；AP 和 STA 同时使用；
- 在 BSS 网络中，支持多个组播网络，并且支持各个组播网络加密方式不同，最多可以支持总和为 32 个的组播网络和入网 STA 加密；
- BSS 网络支持做为 AP 使用时，支持站点与组的总和为 32 个；
- 接收灵敏度：
  - 20MHz MCS7@-71dBm@10%PER；
  - 40MHz MCS7@-67dBm@10%PER；
  - 54Mbps@-73dBm@10%PER；
  - 11Mbps@-86dBm@8%PER；
  - 1Mbps@-96dBm@8%PER；
- 支持多种不同的接收帧过滤选项；
- 支持监听功能；

- 蓝牙协议与功能

- 集成蓝牙基带处理器/协处理器，支持 BT/BLE4.2 协议
- 支持 DR/EDR 各种速率；
- 支持 BLE 1Mbps 速率；

- 供电与功耗
  - 3.3V 单电源供电；
  - 支持 Wi-Fi 节能模式功耗管理；
  - 支持工作、睡眠、待机、关机工作模式；
  - 待机功耗小于 15uA；

3 概述

本芯片是一款支持多接口、多协议的无线局域网 802.11n（1T1R）的 SOC 芯片。该 SOC 芯片集成射频收发前端 RF Transceiver，CMOS PA 功率放大器，基带处理器/媒体访问控制，SDIO、SPI、UART、GPIO 等接口的低功耗 WLAN 芯片。

W800 芯片支持 GB15629.11-2006、IEEE802.11 b/g/ n 协议，并且支持 STBC、Green Field、Short-GI、反向传输、RIFS 帧间隔、AMPDU、AMSDU、T-immediate Compressed Block Ack、normal ACK、no ACK、CTS to self 等丰富的协议以及操作。

W800 芯片片内集成了射频收发前端、A/D 和 D/A 转换器。它支持 DSSS（直接序列扩频）以及 OFDM（正交频分复用）调制模式，具备数据解扰能力，支持多种不同的数据传输速率。在收发器的模拟前端配备的收发 AGC 功能使得芯片系统能够获得最佳的性能。W800 芯片还包含了内置增强信号监测器，可以很大程度上消除多径效应的影响。

W800 芯片在安全方面，不仅支持国家标准的 WAPI 加密，还支持国际标准 WEP、TKIP、CCMP 加密，这些硬件组件使得基于该芯片的数据传输系统在进行保密通信时仍然能够获得与非加密通信时相近的数据传输性能。

W800 芯片除了支持 IEEE802.11 以及 Wi-Fi 协议规定的节能操作外，还支持用户定制的节能方案。芯片支持工作、睡眠、待机、关机四种工作模式，从而使整个系统实现低功耗，并方便用户根据自身的使用场景定义不同的节能方案。

W800 芯片集成了高性能的 32 位嵌入式处理器，大量的内存资源，以及丰富的外设接口，便于使用者很容易的将芯片应用于特定产品的二次开发工作。

W800 芯片支持 AP 功能，可以实现同时组建 5 个 SSID 网络，实现 5 个独立 AP 的功能。支持建立多组播网络功能。可以实现作为 STA 加入别的网络的同时，自己又作为 AP 建立 BSS 网络的功能。

W800 芯片支持 WPS 方式，从而让用户采用一键式操作即可以实现加密的完全网络，保证信息的安全性。

W800 芯片多功能、高集成度保证了 WLAN 系统不需要过多的片外电路和外部存储器。

4 芯片结构

4.1 芯片结构

下图描述 W800 芯片的整体结构，核心部分包括 XT804 CPU，288KB SRAM 和 20KB ROM 存储空间。PMU 部分作为芯片的常供电模块提供了上电时序管理，起振时钟，实时时钟功能等。提供了丰富的外设功能和硬件加解密功能。Wi-Fi 部分集成了 MAC，BB 和 RF。

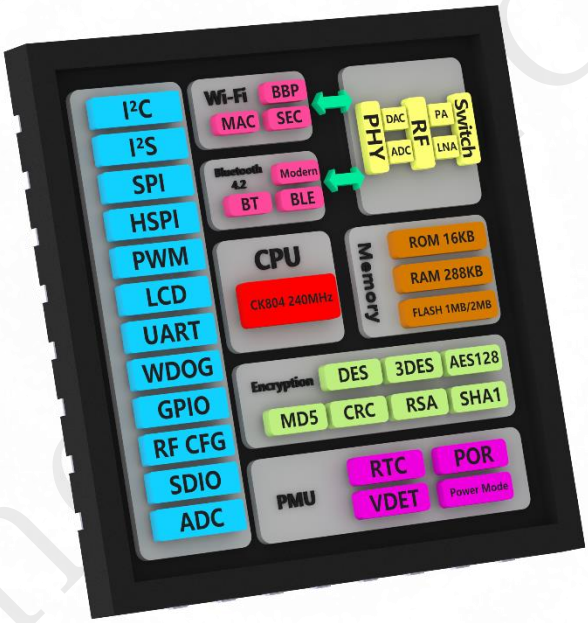


图 1 W800 芯片结构图

4.2 总线结构

W800 芯片由两级总线构成，如下图所示

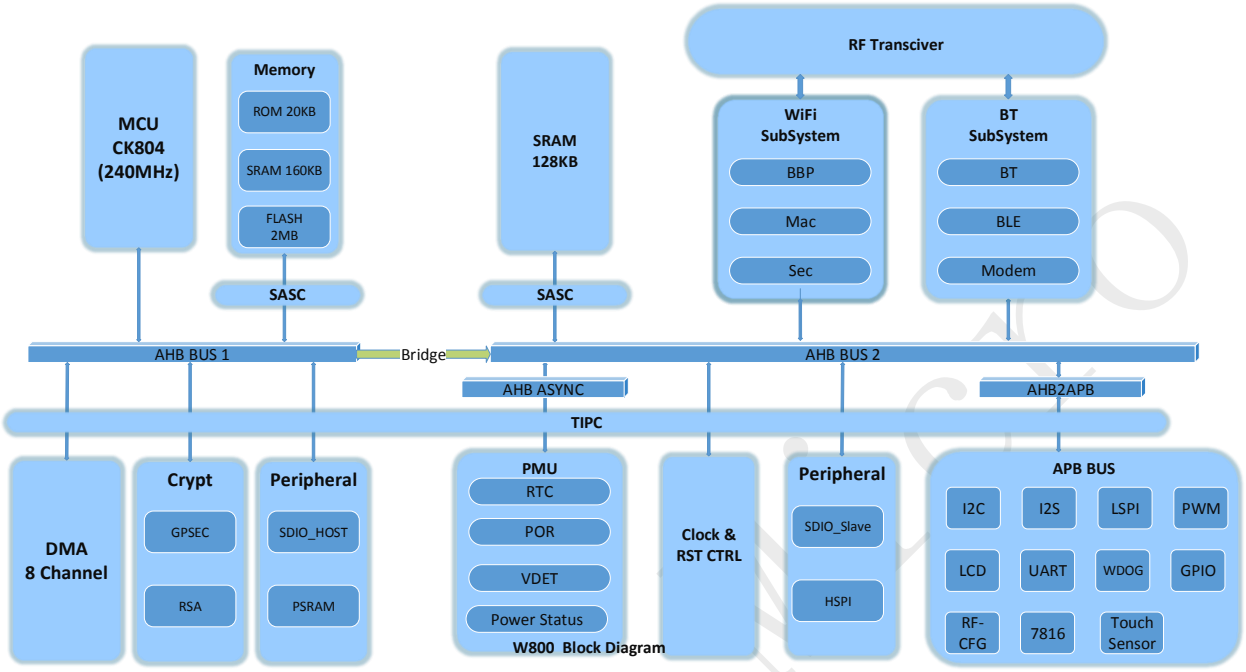


图 2 W800 总线结构图

（1）AHB-1 总线

本级总线有四个主设备-即 XT804，DMA，GPSEC 以及 5 个从设备。

XT804 是面向控制领域的 32 位高能效嵌入式 CPU 核， 采用 16/32 位混合编码指令系统，设计了精简高效的 3 级流水线。

XT804 提供多种可配置功能，包括硬件浮点单元、片上高速缓存、DSP 加速单元、可信防护技术、片上紧耦合 IP 等，用户可根据应用需要进行配置。此外，XT804 提供多总线接口，支持系统总线、指令总线、数据总线的灵活配置。XT804 针对中断响应做了特殊的加速，中断响应延时仅需 13 个周期。总线时钟最快工作在 240MHz 频率，可以配置为 240/160/120/80/40MHz，或更低。

表 1 AHB-1 总线主设备列表

主设备	功能
CPU	完成芯片寄存器配置，存储器管理与使用，以及完整的 802.11MAC 协议。最高运行频率 240MHz
DMA	支持链表结构的独立的 8 通道 DMA 模块，支持片上 16 个硬件 DMA 请求源。
GPSEC	通用加密模块，支持 DES/3DESSHA1/AES/MD5/RC4/CRC/PRDN。自动完成指定内存空间内的数据块加密并回写。

表 2 AHB-1 总线从设备列表

从设备	功能
ROM	ROM 用来存放 CPU 上电后的初始化固件。 主要完成芯片寄存器空间的初始配置等工作。在完成上述工作后，将 CPU 控制权交给 FLASH 中存储的固件。
AHB2AHB	完成 CPU 总线时钟域到 BusMatrix2 总线时钟域主设备访问的转换。要求时钟域必须同源，且 CPU 时钟与 BusMatrix2 时钟频率的比例为 M：1，M 为大于等于 1 的整数。
FLASH	存储固件代码以及运行参数。
SRAM 160KB	可用于存放指令或数据，固件可以根据需要使用该存储器。
RSA	支持最长 2048bit 的 RSA 加解密运算

GPSEC	通用加密/解密模块，支持 SHA1/AES/MD5/RC4/CRC/TRNG。自动完成指定内存空间内的数据块加密/解密并回写。
SDIO_HOST	SDIO 2.0 标准的 SDIO HOST 控制器；可通过此接口访问 SDIO 接口外设。SDIO 接口时钟为总线时钟分频得到，最高支持 50MHz。
PSRAM_CTRL	QSPI 接口的 PSRAM 控制器。可通过此控制器访问外接 PSRAM。QSPI 接口时钟由总线时钟分频得到，最高支持 80MHz 时钟。

（2）AHB-2 总线

本条总线有 4 个主设备，3 个从设备，使用 crossbar 连接结构，能够实现不同主设备对不同从设备的同时访问，从而加大带宽。总线时钟最快工作在 40MHz 频率，可以根据需要配置为更低。

表 3 AHB-2 总线主设备列表

主设备	功能
MAC	802.11MAC 控制协议处理模块。对总线的操作主要包括发送读取数据，接收写数据，以及发送完成描述符的回写等操作。
SEC	安全模块，完成发送接收数据加解密和搬移。发送时，将发送数据和 MAC 描述符搬移至指定位置，并完成加密；接收时，将接收数据和 MAC 接收描述符搬移至指定位置，并完成解密。
AHB2AHB	AHB-1 总线到 AHB-2 总线的总线主设备访问的转换。
SDIO/HSPI	将主机通过 SDIO2.0 设备控制器或高速 SPI 从设备控制器对芯片的访问转换为 AHB 总线信号，并访问内容存储器和寄存器空间。

各主设备采用固定优先级，自上而下优先级递减。

表 4 AHB-2 总线从设备列表

从设备	功能
SRAM 128KB	用于存储上行和下行数据缓存，SDIO/SPI/UART 接口使用此 RAM 用作数据缓存
Configuration	寄存器配置空间，高速模块配置寄存器在这里统一编址。
APB	所有低速模块访问空间，各种低速模块使用 APB 总线连接。
BT_CORE	蓝牙控制器。

4.3 时钟结构

W800 使用 24/40MHz 晶体作为 SoC 时钟源，片内内置 1 个 DPLL 输出 480MHz，供给 CPU，系统总线，数据总线及 WiFi 系统使用；片内另外内置 32.768KHZ RC 振荡器，供 PMU 及 LCD 模块使用。时钟结构概括图如下图所示。

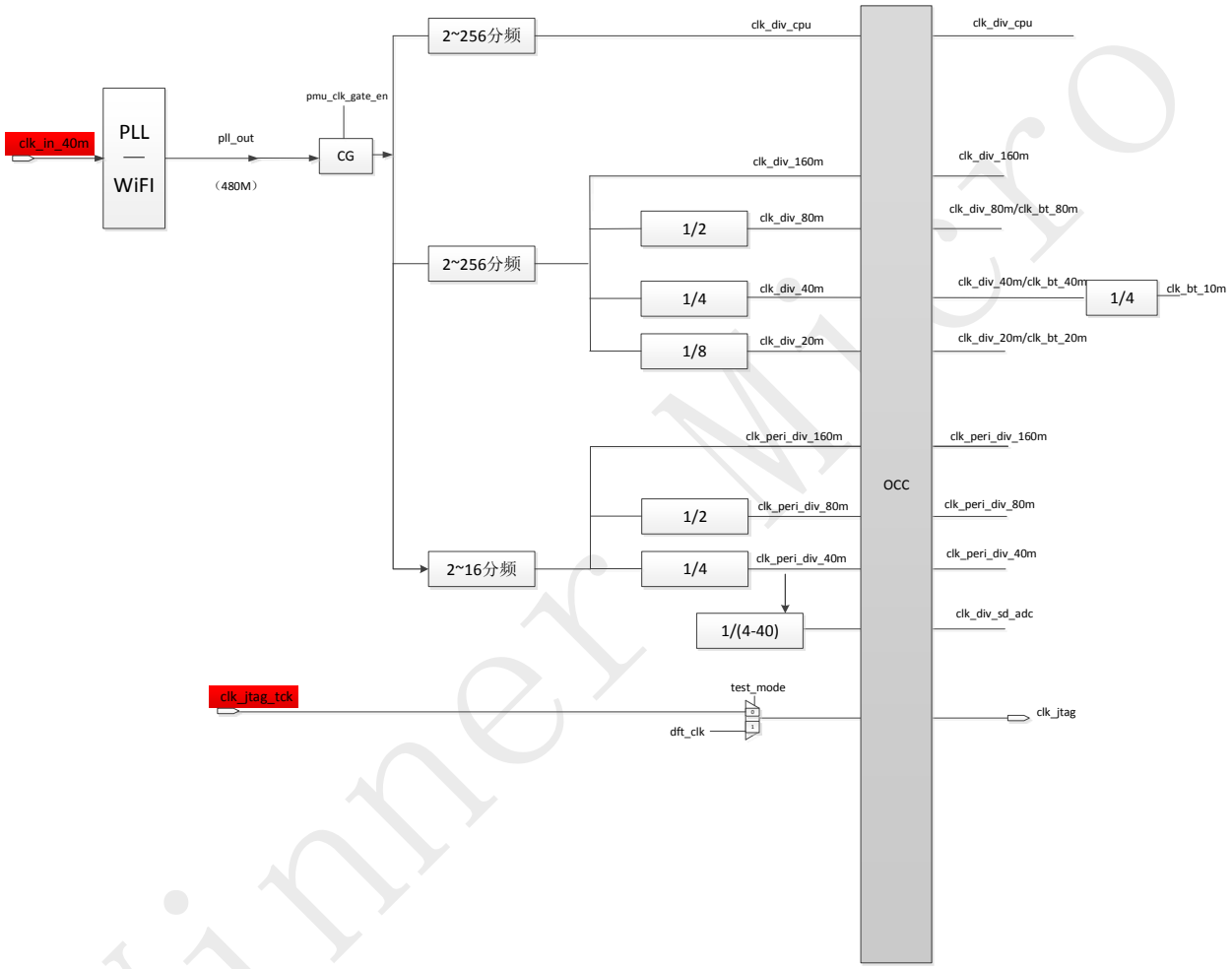
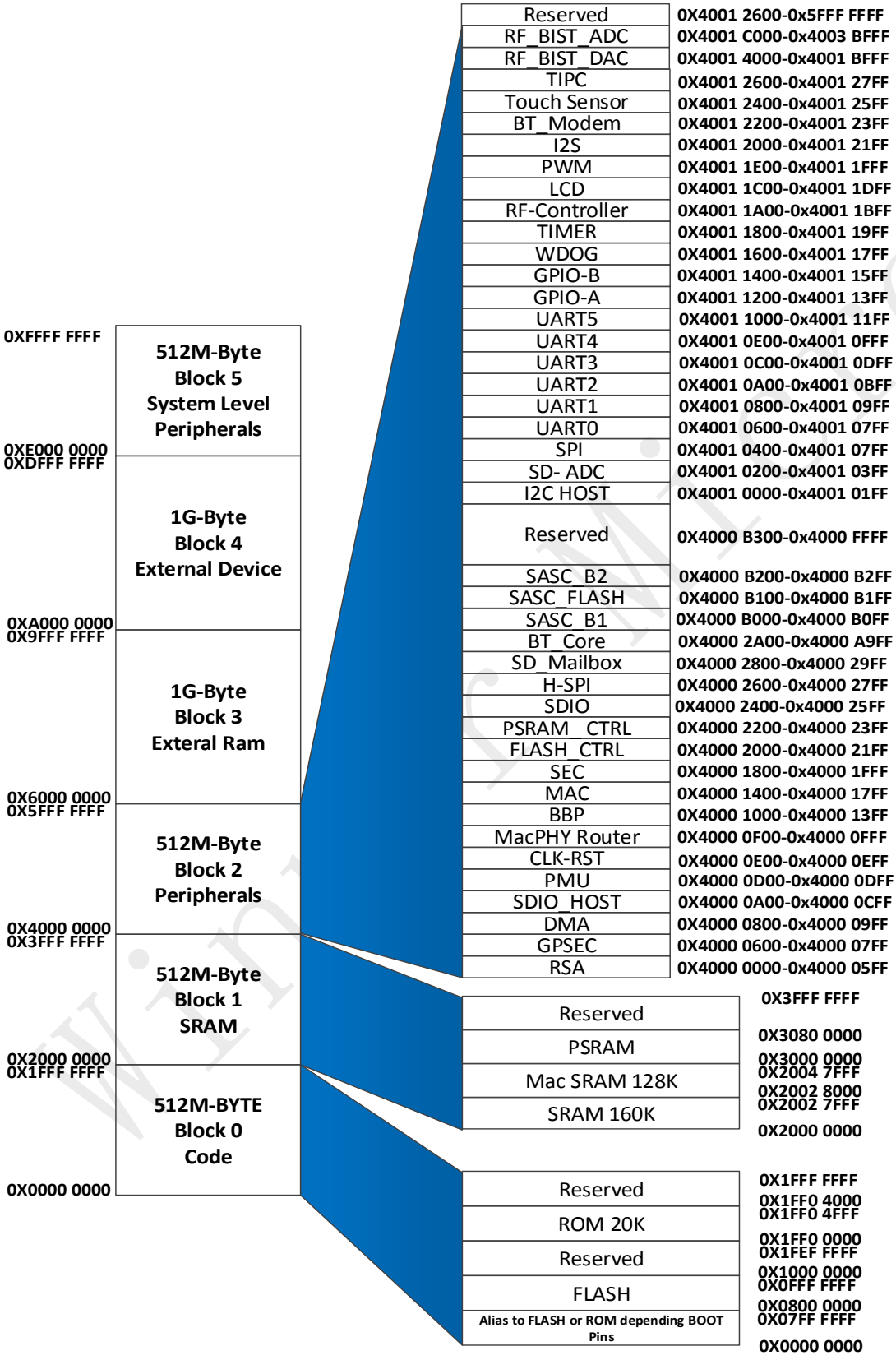


图 3 W800 时钟结构

4.4 地址空间



W800 地址映射

XT804 支持 4G 存储空间，如上图所示分为 6 个 block，分别为代码区，内存区，片上外设，片外存储区，片外外设和系统外设区。根据需求，w800 片内存储空间如图三所示映射到前三区。

表 5　　总线设备地址空间详细划分

总线从设备	BootMode=0	地址空间细分	备注
ROM	**0x0000 0000 ~ 0x0004 FFFF**		存放固化的固件代码
FLASH	**0x0800 0000 ~ 0x0FFF FFFF**		作为专用的指令存储器。
SRAM	0x2000 0000 ~ 0x2002 7FFF		固件内存和指令存储区
Mac RAM	0x2002 8000 ~ 0x2004 7FFF		SDIO/H-SPI/UART 数据缓存
PSRAM	0x3000 000~0x30800000		外设内存
CONFIG	0x4000 0000 ~ 0x4000 2FFF	0x4000 0000 ~ 0x4000 05FF	RSA 配置空间
		0x4000 0600 ~ 0x4000 07FF	GPSEC 配置空间
		0x4000 0800 ~ 0x4000 09FF	DMA 配置空间
		0x4000 0A00 ~ 0x4000 0CFF	SDIO_HOST 配置空间
		0x4000 0D00 ~ 0x4000 0DFF	PMU 配置空间

		0x4000 0E00 ~ 0x4000 0EFF	Clock 与 Reset 配置空间
		0x4000 0F00 ~ 0x4000 0FFF	MacPHY Router 配置空间
		0x4000 1000 ~ 0x4000 13FF	BBP 配置空间
		0x4000 1400 ~ 0x4000 17FF	MAC 配置空间
		0x4000 1800 ~ 0x4000 1FFF	SEC 配置空间
		0x4000 2000 ~ 0x4000 21FF	FLASH Controller 配置空间
		0x4000 2200 ~ 0x4000 23FF	PSRAM_CTRL 配置空间
		0x4000 2400 ~ 0x4000 25FF	SDIO Slave 配置空间
		0x4000 2600 ~ 0x4000 27FF	H-SPI 配置空间
		0x4000 2800 ~ 0x4000 29FF	SD Wrapper 配置空间
		0x4000 2A00 ~ 0x4000 A9FF	BT Core 配置空间
		0x4000 B000 ~ 0x4000 B0FF	SASC-B1 一级总线内存安全配置模块
		0x4000 B100 ~ 0x4000 B1FF	SASC-Flash Flash 安全配置模块

		0x4000 B200 ~ 0x4000 B2FF	SASC-B2 二级总线内存安全配置模块
APB	0x4001 0000 ~ 0x 4001 C000	0x4001 0000 ~ 0x4001 01FF	I2C master
		0x4001 0200 ~ 0x4001 03FF	Sigma ADC
		0x4001 0400 ~ 0x4001 07FF	SPI master
		0x4001 0600 ~ 0x4001 07FF	UART0
		0x4001 0800 ~ 0x4001 09FF	UART1
		0x4001 0A00 ~ 0x4001 0BFF	UART2
		0x4001 0C00 ~ 0x4001 0DFF	UART3
		0x4001 0E00 ~ 0x4001 0FFF	UART4
		0x4001 1000 ~ 0x4001 11FF	UART5
		0x4001 1200 ~ 0x4001 13FF	GPIO-A
		0x4001 1400 ~ 0x4001 15FF	GPIO-B
		0x4001 1600 ~ 0x4001 17FF	WatchDog
		0x4001 1800 ~ 0x4001 19FF	Timer
		0x4001 1A00 ~ 0x4001 1BFF	RF_Controller
		0x4001 1C00 ~ 0x4001 1DFF	LCD
		0x4001 1E00 ~ 0x4001 1FFF	PWM

		0x4001 2000 ~ 0x4001 22FF	I2S
		0x4001 2200 ~ 0x4001 23FF	BT-modem
		0x4001 2400 ~ 0x4001 25FF	Touch Sensor
		0x4001 2600 ~ 0x4001 25FF	TIPC Interface 安全设置
		0x4001 4000 ~ 0x4000 BFFF	RF_BIST DAC 发射内存
		0x4001 C000 ~ 0x4003 BFFF	RF_BIST ADC 接收内存
		0x4001 3C00 ~ 0x5FFF FFFF	RSV

4.4.1 SRAM

W800 内置 288KB SRAM。其中 160KB 挂载一级 AHB 总线上，128KB 挂载在二级 AHB 总线上。CPU 等一级总线设备可以访问所有内存区域，但是二级总线上的设备只能访问二级总线上 128KB 的内存。

4.4.2 Flash

4.4.2.1 QFlash

W800 内部集成 2MBytes QFlash。通过芯片内部集成 32KB cache 实现 XIP 方式在 QFlash 上执行程序。 程序运行过程中，CPU 首先从 Cache 中读取指令，当不能获取指令时，以 8Bytes 一行的方式从 QFlash 读取指令，存入 Cache 内。因此，当持续运行代码大小小于 32K 时，CPU 将无需从 QFlash 读取指令，此时 CPU 可以运行在更高的频率。上述方式为读取指令操作方式，整个 Image 的 RO 段都会以这种方式操作。此过程用户无需干预。

QFlash 也可以存储数据，当用户程序需要读写 QFlash 内数据时，需要通过内置的 QFlash 控制器进行操作，QFlash 提供了相应的地址、指令等寄存器来协助实现用户想要的操作。具体描述请参考 QFlash 控制器对应章节。

用户需要注意的是，程序进行读取或者写入数据时，无需进行状态判断、等待等操作，因为 QFlash 控制器本身会进行判断。当 QFlash 控制器返回时，表明读取或者写入已经完成。

4.4.2.2 SPI Flash

W800 芯片除了支持 6PIN 的 QFlash 接口之外（内置 PIN，未封装），还支持低速 SPI 接口访问。该 SPI 接口的最高工作频率可达 20MHz，支持主从功能，详细描述参考 SPI 接口对应章节。

4.4.3 PSRAM

W800 内置 SPI/QSPI 接口的 PSRAM 控制器，支持外置最大容量 64Mb 的 PSRAM 设备访问，提供总线方式的 PSRAM 读写擦操作。最高读写速度 80MHz。当存储容量需要扩充时，可以使用片外 PSRAM 扩充代码存储空间或者数据存储空间。PSRAM 同样支持 XIP 方式执行程序，CPU Cache 同样支持缓存 PSRAM 中数据。

4.5 启动配置

W800 芯片上电后，CPU 会启动执行 ROM 中的固件，加载 Flash 中指定地址的用户 Image。

ROM 固件在开始运行时会读取 BootMode(PA0)引脚，根据引脚的信号判断进入启动状态：

表 6 启动配置

BootMode	启动条件	启动模式
高		正常启动流程
低	持续<30ms，快速测试模式无效	正常启动流程
	持续>=30ms	进入功能模式
注： 测试模式：芯片测试功能，用户不可操作。 功能模式：进入 ROM 实现的基本功能，例如：下载固件，烧写 MAC 地址等，详细信息参考		

《WM_W800_ROM 功能简述.pdf》

通常，BootMode 引脚应该用于生产或者调试阶段。在生产阶段，用户通过将 BootMode 引脚持续拉低 30ms 以上，进入功能模式，可以快速进行烧录 Flash 工作。

在产品返工或者维修的场景中，在芯片未进入“最高安全等级”（关于安全等级的描述请参考《WM_W800_ROM 功能简述》）时，可以通过该引脚进入功能模式，擦除旧的 Image，写入新的 Image。

在调试阶段，无论固件出现任何故障，都可以通过将 BootMode 引脚持续拉低 30ms 以上，进入串口下载功能，烧录新的固件。

5 时钟与复位模块

5.1 功能概述

时钟与复位模块完成了软件对芯片时钟和复位系统的控制。时钟控制包括时钟变频，时钟关断以及自适应门控；复位控制包括系统以及子模块的软复位控制。

5.2 主要特性

- 支持各模块时钟关断
- 支持部分模块时钟自适应关断
- 支持各模块软件复位
- 支持 CPU 频率设置
- 支持 ADC/DAC 回环测试
- 支持 I2S 时钟设置

5.3 功能描述

5.3.1 时钟门控

通过配置时钟门控使能寄存器 CLK_GATE_EN 可以控制指定功能的时钟关断，从而达到关断某一模块功能的目的。

为了提供固件对系统功耗控制的灵活性，时钟与复位模块提供了系统内各模块的时钟门控功能。当关闭相应模块的时钟时，该模块的数字逻辑与时钟树将停止工作，能够降低系统的动态功耗。

具体各模块的开关对应寄存器 SW_CLKG_EN 的详细描述。

5.3.2 时钟自适应关断

芯片依据内部的某些状态的迁移，自适应关断某些功能模块的时钟。

用户请不要更改配置，否则可能会在配置 PMU 功能时导致系统异常。

5.3.3 功能复位

芯片提供了各子系统的软复位功能，通过设置 SW_RST_CTRL 相应 BIT 为 0 可以达到子系统复位。

但是，复位状态不会自动清除，要恢复正常工作需将 SW_RST_CTRL 相应 BIT 位置 1。

软复位功能并不会复位 CPU 及 WatchDog。

该寄存器中，对 APB/BUS1/BUS2(对应 APB 总线，系统总线及数据总线)的复位操作不推荐，会导致系统访问设备异常。

5.3.4 时钟分频

W800 系统采用 40MHz/24MHz 晶体作为系统时钟源，系统内置 DPLL，固定输出 480MHz 时钟作为全系统的时钟源（如下图）。

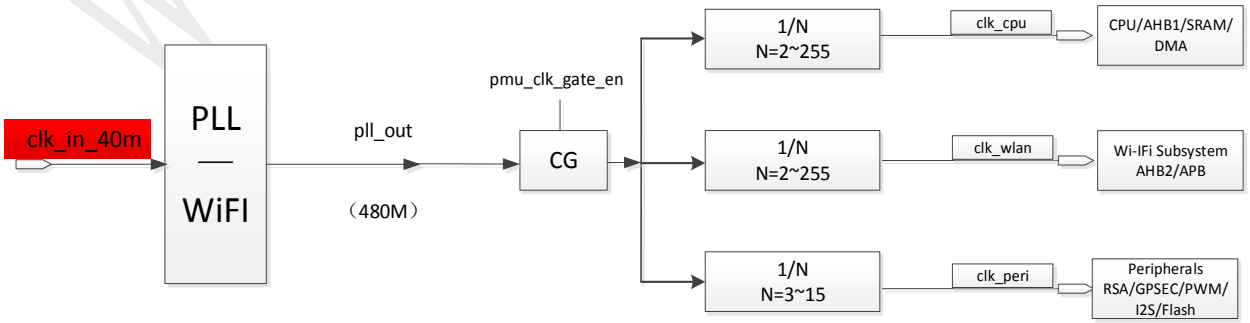


图 4 系统时钟分频关系

系统总线的时钟与 CPU 时钟一致，数据总线的时钟固定为 WLAN 根时钟的 1/4。

WLAN 根时钟同时也是整个 WLAN 系统的时钟源头。

此模块中提供了设定 CPU 时钟与 WLAN 根时钟的功能，供固件调节系统性能及功耗使用。

设置 SYS_CLK_DIV 寄存器的 BIT[7:0]可以调整 CPU 时钟分频系数。CPU 时钟分频的源时钟为 DPLL 的输出，固定为 480MHz。CPU 时钟分频系数默认值为 6，即 CPU 默认工作频率为 480MHz 的 6 分频，即 80MHz。当需要调整 CPU 所需时钟时，可以重新配置本参数。

CLK_PERI 时钟提供 SoC 系统中加密模块的运行时钟的根时钟，以及某些接口的运行时钟的根时钟，比如 PWM 接口，I2S 接口，Flash 接口时钟。此时钟也由 DPLL 输出的 480MHz 分频得出。正常工作情况下应固定为 3 分频，得到 CLK_PERI 根时钟 160MHz。由 CLK_PERI 根时钟进行 2 分频，4 分频得到 80MHz 和 40MHz，提供给加密模块和接口模块使用。

设置 SYS_CLK_DIV 寄存器的 BIT[15:8]可以调整 WLAN 时钟分频系数。默认分频因子为 3，即对 DPLL 的 480MHz 输出 3 分频，得到 160MHz 时钟，作为根节点时钟送给 WLAN（WLAN 再继续分频得到更为详细的低频时钟供 WLAN 系统使用。

注意：如果希望 WLAN 系统正常工作，WLAN 根时钟需要保持在 160MHz，否则 WLAN 系统将失效。当不需要 WLAN 系统工作的时候，可以将 WLAN 根时钟降低，降低系统动态功耗。

在改变系统时钟配置的时候，需要注意：系统总线与数据总线的比例需要维持在 M：1，其中 M 为整数，最小为 1。在改变系统时钟配置时，也需要同时更新寄存器 SYS_CLK_DIV 的 BIT [23:16]，设置正确的比例系数。否则，访问数据总线将得到异常数据。

SYS_CLK_SEL 的[15:8]提供了设置 SAR_ADC 工作频率的分频因子，以 40M 为时钟源进行分频。分频系数即为所配分频值。

SYS_CLK_SEL 的 BIT[4]为配置 RSA 模块核心运算的时钟频率选择，可以选择 80MHz 或者 160MHz。

BIT[5]为配置 GPSEC 模块核心运算的时钟频率选择，可以选择 80MHz 或者 160MHz。

BIT[6]为配置 FLASH 模块对外总线的时钟频率选择，可以选择 40MHz 或者 80MHz。

当需要重新配置 cpu_clk_divider，wlan_clk_divider，bus2_syncdn_factor，sdadc_fdiv 时，需要置位 SYS_CLK_DIV 的 BIT[31]，硬件自动更新上述四个参数到分频器，然后清零 BIT[31]。

I2S_CLK_CTRL 提供了 I2S 模块的时钟配置功能。

5.3.5 调试功能控制

用户可以通过设置 DEBUG_CTRL 的值(SYS_CLK_SEL- BIT[16])来达到使能和禁用 JTAG 功能的目的。

5.4 寄存器描述

5.4.1 寄存器列表

表 7 时钟复位模块寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	软件时钟门控使能寄存器	SW_CLKG_EN	RW	软件配置模块是否关断时钟	0X0000_7FFF
0X0004	软件时钟掩码寄存器	SW_CLK_MASK	RW	软件配置模块是否自适应关断时钟	0X0000_007E
0X0008	保留				
0X000C	软件复位控制寄存器	SW_RST_CTRL	RW	软件配置复位模块	0X01FF_FFFF
0X0010	时钟分频配置寄存器	SYS_CLK_DIV	RW	配置时钟分频比	0X0000_2212
0X0014	调试控制寄存器	DEBUG_CTRL	RW	配置 ADC/DAC 回环测试	0X0000_0000
0X0018	I2S 时钟控制寄存器	I2S_CLK_CTRL	RW	配置 I2S 时钟	0X0000_0000
0X001C	复位状态寄存器	RESET_STATUS	RW	查看 CPU 软复位与 Watchdog 复位状态	0x0000_0000

5.4.2 软件时钟门控使能寄存器

表 8 软件时钟门控使能寄存器

位	访问	操作说明	复位值
[31:22]	RO	保留	
[21]	RW	soft_touch_gate_en 配置 touch_sensor 模块时钟的门控，默认 touch_sensor 模块门控开启 0：touch_sensor 模块时钟关闭	1'b1

		1： touch_sensor 时钟开启	
[20]	RW	soft_bt_gate_en 配置 BT./BLE 模块时钟的门控，默认 BT/BLE 模块门控开启 0：BT/BLE 模块时钟关闭 1： BT/BLE 时钟开启	1'b1
[19]	RW	soft_qsram_gate_en 配置 qspi_ram 模块时钟的门控，默认 qspi_ram 模块门控开启 0：qspi_ram 模块时钟关闭 1： qspi_ram 时钟开启	1'b1
[18]	RW	soft_sdio_m_gate_en 配置 sdio_master 模块时钟的门控，默认 sdio_master 模块门控开启 0：sdio_master 模块时钟关闭 1： sdio_master 时钟开启	1'b1
[17]	RW	soft_gpsec_gate_en 配置 gpsec 模块时钟的门控，默认 gpsec 模块门控开启 0：gpsec 模块时钟关闭 1： gpsec 时钟开启	1'b1
[16]	RW	soft_rsa_gate_en 配置 RSA 时钟的门控，默认 RSA 门控开启 0：RSA 模块时钟关闭 1：RSA 时钟开启	1'b1
[15]	RW	soft_i2s_gate_en	1'b1

		配置 i2s 时钟的门控，默认 i2s 门控开启 0：i2s 时钟关闭 1：i2s 时钟开启	
[14]	RW	soft_lcd_gate_en 配置 lcd 时钟的门控，默认 lcd 门控开启 0：lcd 时钟关闭 1：lcd 时钟开启	1'b1
[13]	RW	Soft_pwm_gate_en 配置 pwm 时钟的门控，默认 pwm 门控开启 0：pwm 时钟关闭 1：pwm 时钟开启	1'b1
[12]	RW	soft_sd_adc_gate_en 配置 sd_adc_时钟的门控，默认 sd_adc_门控开启 0：sd_adc_时钟关闭 1：sd_adc_时钟开启	1'b1
[11]	RW	soft_gpio_gate_en 配置 GPIO 时钟的门控，默认 GPIO 门控开启 0：GPIO 时钟关闭 1：GPIO 时钟开启	1'b1
[10]	RW	soft_timer_gate_en 配置 timer 时钟的门控，默认 timer 门控开启 0：timer 时钟关闭	1'b1

		1：timer 时钟开启	
[9]	RW	soft_rf_cfg_gate_en：**内部使用，请勿修改** 配置 rf_cfg 时钟的门控，默认 rf_cfg 门控开启 1'b0：rf_cfg 时钟关闭 1'b1：rf_cfg 时钟开启	1'b1
[8]	RW	soft_dma_gate_en 表示供给 dma 时钟域的时钟是否关断 1'b0：dma 时钟关闭 1'b1：dma 时钟开启	1'b1
[7]	RW	soft_ls_spi_gate_en 配置低速 spi 时钟的门控，默认低速 spi 门控开启 1'b0：低速 spi 时钟关闭 1'b1：低速 spi 时钟开启	1'b1
[6]	RW	soft_uart5_gate_en 配置 uart5 的门控，默认 uart5 开启 0：uart5 关闭 1：uart5 开启	1'b1
[5]	RW	soft_uart4_gate_en 配置 uart4 的门控，默认 uart4 开启 0：uart4 关闭 1：uart4 开启	1'b1
[4]	RW	soft_uart3_gate_en	1'b1

		配置 uart3 的门控，默认 uart3 开启 0：uart3 关闭 1：uart3 开启	
[3]	RW	soft_uart2_gate_en 配置 uart2 的门控，默认 uart2 开启 0：uart2 关闭 1：uart2 开启	1'b1
[2]	RW	soft_uart1_gate_en 配置 uart1 时钟的门控，默认 uart1 门控开启 1'b0：uart1 时钟关闭 1'b1：uart1 时钟开启	1'b1
[1]	RW	soft_uart0_gate_en 配置 uart0 时钟的门控，默认 uart0 门控开启 1'b0：uart0 时钟关闭 1'b1：uart0 时钟开启	1'b1
[0]	RW	soft_i2c_gate_en 配置 i2c 时钟的门控，默认 i2c 门控开启 1'b0：i2c 时钟关闭 1'b1：i2c 时钟开启	1'b1

5.4.3 软件时钟掩码寄存器

表 9 软件时钟掩码寄存器

位	访问	操作说明	复位值
[6]	RW	soft_cpu_clk_gt_mask 表示供给 CPU 时钟域（包括 CPU，bus1、ROM、SRAM）的时钟是否能够自适应的关断（**当 CPU 需要进入 WFI 状态的时候，不要设置自适应关断**） 1'b0：允许自适应关断和开启 1'b1：不允许自适应关断和开启	1'b1
[5：2]	RW	保留，**内部使用，请勿修改**	
[1]	RW	soft_sdioahb_clk_gt_mask 表示供给 sdio ahb 时钟域的时钟是否能够自适应的关断 1'b0：允许自适应关断和开启 1'b1：不允许自适应关断和开启	1'b1
[0]	RW	soft_pmu_clk_gt_mask pll 输出的时钟后有一个门控单元，采用该寄存器配置，表示是否允许被 PMU 关断。 1'b0：允许 PMU 关断该门控单元，从而关断时钟 1'b1：不允许 PMU 关断该门控单元	1'b0

5.4.4 软件复位控制寄存器

表 10 软件复位控制寄存器

位	访问	操作说明	复位值
[31]	RW	soft_touch_rst_n 软件复位 touch_sensor 模块 0：复位	1'b1

		1：复位释放	
[30]	RW	soft_rst_flash_n 软件复位 Flash 控制器模块 0：复位 1：复位释放	1'b1
[29]	RW	soft_rst_bt_n 软件复位 BT 模块 0：复位 1：复位释放	1'b1
[28]	RW	soft_rst_qspi_ram_n 软件复位 qspi_ram 模块 0：复位 1：复位释放	1'b1
[27]	RW	soft_rst_sdio_m_n 软件复位 sdio_master 模块 0：复位 1：复位释放	1'b1
[26]	RW	soft_rst_gpsec_n 软件复位 gpsec 模块 1'b0：复位 1'b1：复位释放	1'b1
[25]	RW	soft_rst_rsa_n	1'b1

		软件复位 RSA 模块 1'b0：复位 1'b1：复位释放	
[24]	RW	soft_rst_i2s_n 软件复位 i2s 模块 1'b0：复位 1'b1：复位释放	1'b1
[23]	RW	soft_rst_lcd_n 软件复位 lcd 模块 1'b0：复位 1'b1：复位释放	1'b1
[22]	RW	soft_rst_pwm_n 软件复位 pwm 模块 1'b0：复位 1'b1：复位释放	1'b1
[21]	RW	soft_rst_sar_adc_n 软件复位 sar_adc 模块 1'b0：复位 1'b1：复位释放	1'b1
[20]	RW	soft_rst_timer_n 软件复位 timer 模块 1'b0：复位	1'b1

		1’b1：复位释放	
[19]	RW	soft_rst_gpio_n 软件复位 gpio 模块 1’b0：复位 1’b1：复位释放	1’b1
[18]	RW	soft_rst_rf_cfg_n 软件复位配置 RF 的寄存器模块(**内部使用，请勿修改)** 1’b0：复位 1’b1：复位释放	1’b1
[17]	RW	soft_rst_spis_n 软件复位高速 spi 模块 1’b0：复位 1’b1：复位释放	1’b1
[16]	RW	soft_rst_spim_n 软件复位低速 spi 模块 1’b0：复位 1’b1：复位释放	1’b1
[15]	RW	soft_rst_uart5_n 软件复位片内 uart5 模块 1’b0：复位 1’b1：复位释放	1’b1
[14]	RW	soft_rst_uart4_n	1’b1

		软件复位片内 uart4 模块 1'b0：复位 1'b1：复位释放	
[13]	RW	soft_rst_uart3_n 软件复位片内 uart3 模块 1'b0：复位 1'b1：复位释放	1'b1
[12]	RW	soft_rst_uart2_n 软件复位片内 uart2 模块 1'b0：复位 1'b1：复位释放	1'b1
[11]	RW	soft_rst_uart1_n 软件复位片内 uart1 模块 1'b0：复位 1'b1：复位释放	1'b1
[10]	RW	soft_rst_uart0_n 软件复位片内 uart0 模块 1'b0：复位 1'b1：复位释放	1'b1
[9]	RW	soft_rst_i2c_n 软件复位片内 i2c 模块 1'b0：复位	1'b1

		1’b1：复位释放	
[8]	RW	soft_rst_bus2_n 软件复位片内 bus2 模块 1’b0：复位 1’b1：复位释放	1’b1
[7]	RW	soft_rst_bus1_n 软件复位片内 bus1 模块 1’b0：复位 1’b1：复位释放	1’b1
[6]	RW	soft_rst_apb_n 软件复位 abp 桥接模块 1’b0：复位 1’b1：复位释放	1’b1
[5]	RW	soft_rst_mem_mng_n 软件复位 mem_mng 模块(**内部使用，请勿修改)** 1’b0：复位 1’b1：复位释放	1’b1
[4]	RW	soft_rst_dma_n 软件复位 dma 模块 1’b0：复位 1’b1：复位释放	1’b1
[3]	RW	soft_rst_sdio_ahb_n	1’b1

		软件复位 sdio ahb 时钟域模块 1'b0：复位 1'b1：复位释放	
[2]	RW	soft_rst_sec_n 软件复位安全模块(**内部使用，请勿修改**) 1'b0：复位 1'b1：复位释放	1'b1
[1]	RW	soft_rst_mac_n 软件复位 mac 模块(**内部使用，请勿修改**) 1'b0：复位 1'b1：复位释放	1'b1
[0]	RW	soft_rst_bbp_n 软件复位 bbp 模块(**内部使用，请勿修改**) 1'b0：复位 1'b1：复位释放	1'b1

5.4.5 时钟分频配置寄存器

表 11 时钟分频配置寄存器

位	访问	操作说明	复位值
[31]	RW	divide_freq_en 当需要重新配置 cpu_clk_divider ，wlan_clk_divider ，bus2_syncdn_factor， sdadc_fdiv 时，置位本寄存器，硬件自动更新上述四个参数到分频器，然后清零本	1'b0

		寄存器。 1'b0：分频系数已生效 1'b1：要求硬件更新分频参数 注：此处配置分频因子时，当 Divide_freq_en 有效时，所有因子必须已经有效	
[30:28]		保留	
[27:24]	RW	Peripheral_divider 160M 时钟 分频因子： 以 DPLL 为时钟源进行分频。分频系数即为所配分频值。分频输出应为 160MHz。 DPLL 输出为 480MHz，应配置为 3.	4'h3
[23: 16]	RW	bus2_syncdn_factor bus1 和 bus2 的时钟比例关系，应该为 N：1 其中 N 为整数，在实际调整时，主要看 CPU 的工作频率和 bus2 的时钟频率之比。 由于默认 cpu 采用 80MHz 时钟，bus2 采用 40MHz 时钟，则 N=2	8'h2
[15 : 8]	RW	wlan_clk_divider 从 PLL 出来的时钟分频后，送给 wlan 系统。本寄存器为分频因子，该因子>=2。 默认分频因子为 3，即对 pll 的 480MHz 输出 3 分频，得到 160MHz 时钟，作为根节点时钟送给 wlan（wlan 在继续分频得到更为详细的低频时钟）； 注 1：如需 WLAN 系统正常工作，此时钟需固定在 160MHz；如 WLAN 系统关闭，则此时钟可降频节省功耗。此时钟不得配置高于 160MHz。 注 2：二级总线时钟及 APB 时钟为此时钟四分频；	8'h3

[7 : 0]	RW	cpu_clk_divider 从 PLL 出来的时钟分频后，送给 CPU。本寄存器为分频因子，该因子>=2。 默认分频因子为 6，即复位释放后，对 PLL 输出的 480MHz 时钟 6 分频，送给 cpu 的是 80MHz 时钟。当需要调整 cpu 所需时钟时，可以重新配置本参数	8'h6

5.4.6 调试控制寄存器

表 12 时钟选择寄存器

位	访问	操作说明	复位值
[16]	RW	JTAG 使能 1'b0: 禁止 JTAG 调试功能 1'b1: 使能 JTAG 调试功能	1'b0
[15: 8]	RW	sd_adc_div sigma-delta ADC 时钟 分频因子: 以 40MHz 为时钟源进行分频。分频系数即为所配分频值。 配置此寄存器后必须配置寄存器 clk_divider 中 Divide_freq_en 才能生效;	8'd10
[7]	RW	RSV	1'b0
[6]	RW	qflash_clk_sel QSPI_FLASH 时钟选择 1: 使用 80MHz ;	1'b0

		0：使用 40MHz；	
[5]	RW	gpsec_sel GPSEC 时钟选择 1：使用 160MHz ； 0：使用 80MHz；	1’b0
[4]	RW	rsa_sel RSA 时钟选择 1：使用 160MHz ； 0：使用 80MHz；	1’b0
[3:0]	RW	保留，**请勿修改**	4’d0

5.4.7 I2S 时钟控制寄存器

表 13 I2S 时钟控制寄存器

位	访问	操作说明	复位值
[31:18]		保留	
[17: 8]	RW	BCLKDIV BCLK 分配器：F_BCLK = F_I2SCLK / BCLKDIV 注意：如果未选择 EXTAL_EN 而使用内部 PLL 则 F_I2SCLK =F_CPU (与 CPU 频率相同)。 假设 F_CPU = 160MHz，启用 WXTAL_EN 时 F_I2SCLK = 外部晶振频率， **BCLKDIV = round (F_I2SCLK/(Fs*W*F))** 其中 Fs 是音频数据的采样频率，W 是字宽；	10’b0

		数据为单声道时 F = 1； 当数据为立体声时 F = 2。 例如，如果使用内部 PLL 且数据宽度为 24 位，则格式为立体声格式，采样频率为 128KHz，BCLKDIV 应配置为（160 * 10e6 / 128 * 10e3 * 24 * 2）= 10'h1a。	
[7 : 2]	RW	MCLKDIV 如果选择外部时钟，则该 MCLK 分频器用于产生适当的 MCLK 频率。 F_mclk = F_I2SCLK / (2 * MCLKDIV)； 当 MCLKDIV = 0 时 F_I2SCLK 是外部时钟； 当 MCLKDIV >= 1 时 F_mclk = F_I2SCLK； 注意：F_mclk 应配置为 256 * fs，其中 fs 是采样频率。	6'b0
[1]	RW	MCLKEN MCLK 使能开关 1'b0：禁止 MCLK 1'b1：使能 MCLK	1'b0
[0]	RW	EXTAL_EN 外部时钟选择，选择使用内部 I2S 块时钟还是外部时钟 1'b0：内部时钟 1'b1：外部时钟 注意：使用外部时钟时，外部时钟必须为 2 * N * 256 fs，其中 fs 为采样频率，N 必须为整数。	1'b0

5.4.8 复位状态寄存器

表 14 复位状态寄存器

位	访问	操作说明	复位值
[31:18]		保留	
[17: 8]	WO	CPU 软复位状态清除 写 1 清除 CPU soft Reset Status 状态。	1'b0
[7 : 2]	WO	Wdog 软复位状态清除 写 1 清除 Wdog Reset Status 状态。	1'b0
[1]	RO	CPU 软复位 状态 1: CPU 产生过软复位; 0: CPU 未产生软复位;	1'b0
[0]	RO	Wdog 复位状态 1: Wdog 产生 了 Reset; 0: Wdpg 未产生 Reset	1'b0

6 DMA 模块

6.1 功能概述

DMA 用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。可以在无需任何 CPU 操作的情况下通过 DMA 快速移动数据。这样节省的 CPU 资源，不影响 CPU 进行其他指令的操作。

DMA 挂载在 AHB 总线上，最多支持 8 通道，16 个硬件外设请求源，支持链表结构与寄存器控制。

6.2 主要特性

- Amba2.0 标准总线接口，8 路 DMA 通道
- 支持基于存储器链表结构的 DMA 操作
- 支持 16 个硬件外设请求源
- 支持 1，4-burst 操作模式
- 支持 byte、half-word，word 为单位传输操作
- 支持源、目的地址不变或顺序递增或可配置在预定义地址范围内循环操作
- 支持内存到内存，内存到外设，外设到内存的数据传输方式

6.3 功能描述

6.3.1 DMA 通道

W800 共支持 8 路 DMA 通道，DMA 通道互相不干涉，可以同时运行。请求不同的数据流可以选择不同的 DMA 通道。

每个 DMA 通道分配在不同的寄存器地址偏移段，可以直接选择相应通道的地址段进行配置使用即可。不

同通道的寄存器配置方式完全一致。

表 15 DMA 地址分配

DMA 基地址	0x4000 0800
DMA_CH0	偏移量（0x10~0x38）
DMA_CH1	偏移量（0x40~0x68）
DMA_CH2	偏移量（0x70~0x98）
…	…
DMA_CH7	偏移量（0x160~0x188）

6.3.2 DMA 数据流

8 路 DMA 通道能够实现源和目的之间单向数据传输链路。

DMA 的源和目的地址可以设置为每次 DMA 操作完成之后不变、递增或循环三种模式：

- DMA_CTRL[2:1]控制源地址每次 DMA 操作后变化方式；
- DMA_CTRL[4:3]控制目的地址每次 DMA 操作后变化方式。

DMA 可以设置 byte、half-word、word 的搬运单位，最终搬运数据的数量是搬运单位的整数倍，通过 DMA_CTRL[6:5]来设置。

DMA 可以通过 burst 设置每次搬运多少个单位的数据，通过 DMA_CTRL[7]来选择一次搬运 1 或 4 个单位的数据，如果 DMA_CTRL[6:5]设置为 word，burst 设置为 4，则每次搬运 4 个 word 的数据。

DMA 可以设置每次启动 DMA 传输的 Byte 个数，最大 65535 Byte，通过 DMA_CTRL[23:8]来设置。

6.3.3 DMA 循环模式

DMA 循环地址模式是指设置 DMA 的源和目的地址之后，数据搬运达到设定的循环边界之后，会跳转到循环起始地址，如此循环执行，直到到达设定的传输字节。

循环地址模式的源和目的地址需要用 SRC_WRAP_ADDR 和 DEST_WRAP_ADDR 寄存器来设定，并通过 WRAP_SIZE 来设定循环的长度值。

6.3.4 DMA 传输模式

DMA 支持 3 种传输模式：

- 内存到内存

  源地址和目的地址均配置成需要传输的内存地址，DMA_MODE[0]设置为 0，软件方式。

- 内存到外设

  源地址设置为内存地址，目的地址设置成外设地址，DMA_MODE[0]设置为 1，硬件方式，DMA_MODE[5:2]选择所使用的外设。

- 外设到内存

  源地址设置为外设地址，目的地址设置成内存地址，DMA_MODE[0]设置为 1，硬件方式，DMA_MODE[5:2]选择所使用的外设。

6.3.5 DMA 外设选择

当使用外设到内存或者内存到外设这种传输方式的时候,除了相应的外设需要设置为 DMA TX 或 RX 外，DMA_MODE[5:2]也需要选择对应的外设。

注意：因为 UART 口共有 3 个，在 UART 使用 DMA 的时候，还需要通过 UART_CH[1:0]来选择对应的 UART。

6.3.6 DMA 链表模式

DMA 支持链表工作模式。 通过链表模式，我们在 DMA 搬运当前链表内存数据的时候，可以提前向下一个链表中填充数据，DMA 搬完当前链表之后，判断到下一个链表有效，可以直接搬运下一个链表的数据。通过链表的方式可以有效的提高 DMA 和 CPU 配合的效率。

链表操作方式：通过 DMA_MODE[1]寄存器设置 DMA 为链表工作方式，再将 DESC_ADDR 寄存器设置为链表结构的起始地址，然后再通过 CHNL_CTRL 寄存器使能 DMA。当 DMA 处理完成当前内存的搬移后，软件通过设置有效标志，通知 DMA 链表中依然存在有效的数据，DMA 依据链表的有效标志处理下一个待搬移数据。

6.3.7 DMA 中断

DMA 传输完成或者 burst 均可以产生中断，INT_MASK 寄存器可以屏蔽 DMA 通道对应的中断。

当 DMA 相应中断产生后，可以通过 INT_SRC 寄存器查询当前中断的状态，指示当前是什么产生的中断，相应的状态位需要软件写 1 清 0。

6.4 寄存器描述

6.4.1 寄存器列表

表 16 DMA 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	中断屏蔽寄存器	INT_MASK	RW	设置需要屏蔽的 DMA 中断	0X0000_FFFF
0X0004	中断状态寄存器	INT_SRC	RW	指示当前 DMA 的中断状态	0X0000_0000
0X0008	DMA channel 选择寄存器	DMA_CH	RW	UART 外设时选择哪个 UART	0X0000_0000

0X000C	保留				
DMA CHNL0 registers					
0X0010	DMA 源地址寄存器	SRC_ADDR	RW	DMA 传输的源地址	0X0000_0000
0X0014	DMA 目的地址寄存器	DEST_ADDR	RW	DMA 传输的目的地址	0X0000_0000
0X0018	DMA 循环源起始地址寄存器	SRC_WRAP_ADDR	RW	循环模式下 DMA 传输源地址	0X0000_0000
0X001C	DMA 循环目的起始地址寄存器	DEST_WRAP_ADDR	RW	循环模式下 DMA 传输目的地址	0X0000_0000
0X0020	DMA 循环长度寄存器	WRAP_SIZE	RW	循环模式下 DMA 循环边界	0X0000_0000
0X0024	DMA 通道控制寄存器	CHNL_CTRL	RW	当前通道 DMA 启动和停止	0X0000_0000
0X0028	DMA 模式选择寄存器	DMA_MODE	RW	设置 DMA 的工作方式	0X0000_0000
0X002C	DMA 数据流控制寄存器	DMA_CTRL	RW	设置 DMA 传输数据流	0X0000_0000
0X0030	DMA 传输字节数寄存器	DMA_STATUS	RO	获取当前已传输的字节数	0X0000_0000
0X0034	DMA 链表入口地址寄存器	DESC_ADDR	RW	DMA 链表地址入口地址设置	0X0000_0000
0X0038	DMA 当前目的地址寄存器	CUR_DEST_ADDR	RO	当前 DMA 操作的地址	0X0000_0000
DMA CHNL1 registers					
0X0040 - 0X0068	同 DMA CHNL0 registers				
DMA CHNL2 registers					
0X0070 - 0X0098	同 DMA CHNL0 registers				
DMA CHNL3 registers					
0X00A0 -	同 DMA CHNL0 registers				

0X00C8	
DMA CHNL4 registers	
0X00D0 - 0X00F8	同 DMA CHNL0 registers
DMA CHNL5 registers	
0X0100 - 0X0128	同 DMA CHNL0 registers
DMA CHNL6 registers	
0X0130 - 0X0158	同 DMA CHNL0 registers
DMA CHNL7 registers	
0X0160 - 0X0188	同 DMA CHNL0 registers

6.4.2 中断屏蔽寄存器

表 17 DMA 中断屏蔽寄存器

位	访问	操作说明	复位值
[31:16]		保留	
[15]	RW	channel7 transfer_done 中断屏蔽，高有效。	1'b1
[14]	RW	channel7 burst_done 中断屏蔽，高有效。	1'b1
[13]	RW	channel6 transfer_done 中断屏蔽，高有效。	1'b1
[12]	RW	channel6 burst_done 中断屏蔽，高有效。	1'b1

[11]	RW	channel5 transfer_done 中断屏蔽，高有效。	1'b1
[10]	RW	channel5 burst_done 中断屏蔽，高有效。	1'b1
[9]	RW	channel4 transfer_done 中断屏蔽，高有效。	1'b1
[8]	RW	channel4 burst_done 中断屏蔽，高有效。	1'b1
[7]	RW	channel3 transfer_done 中断屏蔽，高有效。	1'b1
[6]	RW	channel3 burst_done 中断屏蔽，高有效。	1'b1
[5]	RW	channel2 transfer_done 中断屏蔽，高有效。	1'b1
[4]	RW	channel2 burst_done 中断屏蔽，高有效。	1'b1
[3]	RW	channel1 transfer_done 中断屏蔽，高有效。	1'b1
[2]	RW	channel1 burst_done 中断屏蔽，高有效。	1'b1
[1]	RW	channel0 transfer_done 中断屏蔽，高有效。	1'b1
[0]	RW	channel0 burst_done 中断屏蔽，高有效。	1'b1

6.4.3 中断状态寄存器

表 18 DMA 中断状态寄存器

位	访问	操作说明	复位值
[31:16]		保留	
[15]	RW	channel7 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[14]	RW	channel7 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[13]	RW	channel6 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[12]	RW	channel6 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[11]	RW	channel5 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0

[10]	RW	channel5 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[9]	RW	channel4 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[8]	RW	channel4 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[7]	RW	channel3 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[6]	RW	channel3 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[5]	RW	channel2 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[4]	RW	channel2 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[3]	RW	channel1 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[2]	RW	channel1 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0
[1]	RW	channel0 transfer_done 中断状态，写 1 清 0。DMA 传输完成产生中断。	1'b0
[0]	RW	channel0 burst_done 中断状态，写 1 清 0。DMA burst 完成产生中断。	1'b0

6.4.4 UART 选择寄存器

表 19 UART 选择寄存器

位	访问	操作说明	复位值
[31: 24]		保留	
[23:8]	RW	dma req clear: 对每 bit 写 1 清除 对应的 dma req 请求。自清。 如 对 bit 23 写 1 将清除 dma_sel 中第 15 个 对应的 dma 请求; 对 bit 8 写 1 将清除 dma_sel 中第 0 个 dma 请求-uart_rx_req;	16'd0
[2 : 0]	RW	Uart dma channel 选择: 3'd0: uart0 模块 dma 通道 接入 dma	3'h0

		3'd1: uart1 模块 dma 通道 接入 dma 3'd2: uart2/7816 模块 dma 通道 接入 dma 3'd3: uart3 模块 dma 通道 接入 dma 3'd4: uart4 模块 dma 通道 接入 dma 3'd5: uart5 模块 dma 通道 接入 dma	

6.4.5 DMA 源地址寄存器

表 20 DMA 源地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	非循环模式下，DMA 搬运的源地址，外设地址或内存地址	32'h0

6.4.6 DMA 目的地址寄存器

表 21 DMA 目的地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	非循环模式下，DMA 搬运的目的地址，外设地址或内存地址	32'h0

6.4.7 DMA 循环源起始地址寄存器

表 22 DMA 循环源起始地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	循环模式下，DMA 搬运的源地址的起始地址，外设地址或内存地址	32'h0

6.4.8 DMA 循环目的起始地址寄存器

表 23 DMA 循环目的起始地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	循环模式下，DMA 搬运的目的地址的起始地址，外设地址或内存地址	32'h0

6.4.9 DMA 循环长度寄存器

表 24 DMA 循环长度寄存器

位	访问	操作说明	复位值
[31:16]	RW	循环模式下，DMA 目的地址循环长度。 DMA 从起始地址依次递增搬运数据，当搬运数据字节数达到此设定值之后，会跳转到循环起始地址，从起始地址开始继续搬运数据	16'h0
[15: 0]	RW	循环模式下，DMA 源地址循环长度。	16'h0

6.4.10 DMA 通道控制寄存器

表 25 DMA 通道控制寄存器

位	访问	操作说明	复位值
[31: 2]		保留	
[1]	RW	dma_stop 停止 dma 操作，高有效。 DMA 会在完成当前 burst 操作后停止，并同时清除 chnl_on。软件应根据 chnl_on 为 0 确定 dma 已经完全停止。	1'b0
[0]	RW	chnl_on	1'b0

		启动当前通道 DMA 转换，高有效。 Dma 完成后转换或设置停止后，自动清 0。	

6.4.11 DMA 模式选择寄存器

表 26 DMA 模式选择寄存器

位	访问	操作说明	复位值
[31: 7]		保留	
[6]	RW	chain_link_en 链表方式下有效，表示 dma 在处理完第一个链表后，是否继续读取和处理后续链表。 如果为 1，则更新链表中的 next_desc_addr，并继续读取下一个链表，直到链表中 vld 为 0；如果为 0，则处理完成当前链表后即停止。	1'b0
[5 : 2]	RW	dma_sel 16 个 dma_req 的选择。 4'd0：uart rx dma req 4'd1：uart tx dma req 4'd2：pwm_cap0_req 4'd3：pwm_cap1_req 4'd4：LS_SPI rx dma req 4'd5：LS_SPI tx dma req 4'd6：SD_ADC chnl0 req 4'd7：SD_ADC chnl1 req	4'h0

		4'd8：SD_ADC chnl2 req 4'd9：SD_ADC chnl3 req 4'd10： I2S RX req 4'd11： I2S TX req 4'd12： SDIO_HOST req	
[1]	RW	chain_mode 1'b0：使用普通模式 1'b1：使用链表模式	1'b0
[0]	RW	dma_mode 1'b0：软件方式。 1'b1：硬件方式。	1'b0

6.4.12 DMA 数据流控制寄存器

表 27 DMA 数据流控制寄存器

位	访问	操作说明	复位值
[31:24]		保留	
[23: 8]	RW	total_byte 总共需要操作的 byte 个数。需要与 data_size 配置相一致，即若果为字操作，则应该为配置为 4 的整数倍；如果为半字操作，则应该配置为 2 的整数倍。	16'h0
[7]	RW	burst_size 设置 DMA 每次搬运多少个单位的数据 1'b0：burst 为 1	1'b0

		1'b1：burst 为 4 当最后一次 burst 大小超过剩余传输的个数时，使用 burst 大小为剩余数据的大小。	
[6 : 5]	RW	data_size 设置 DMA 的搬运单位 2'h0：byte 2'h1：half_word 2'h2：word 2'h3：保留	2'h0
[4 : 3]	RW	dest_addr_inc 2'h0：每次操作后目的地址不变； 2'h1：每次操作后目的地址自动累加。 2'h2：保留 2'h3：循环操作，每次操作后目的地址自动累加,到达定义的循环边界跳转到循环起始地址。	2'h0
[2 : 1]	RW	src_addr_inc 2'h0：每次操作后源地址不变； 2'h1：每次操作后源地址自动累加。 2'h2：保留 2'h3：循环操作，每次操作后源地址自动累加,到达定义的循环边界跳转到循环起始地址。	2'h0
[0]	RW	auto_reload	1'b0

		当完成当前 DMA 搬运后，自动按当前 DMA 配置重新下一次 DMA 搬运。	

6.4.13 DMA 传输字节数寄存器

表 28 DMA 传输字节数寄存器

位	访问	操作说明	复位值
[31:16]		保留	
[15: 0]	RW	transfer_cnt 当前传输的字节个数。 每次重新开启 dma（chnl_on 置 1）清 0，并重新开始计数。	16'h0

6.4.14 DMA 链表入口地址寄存器

表 29 DMA 链表入口地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	desc_addr 链表使能时，作为链表的入口地址。每传输完成一次链表后，将下一个链表的基地址更新至此寄存器。	32'h0

6.4.15 DMA 当前目的地址寄存器

表 30 DMA 当前目的地址寄存器

位	访问	操作说明	复位值
[31: 0]	RO	current_dest_addr 当前 DMA 操作目的地址。	32'h0

		当软件停止 dma 时，可以通过查看此寄存器获悉 dma 将要操作的目的地址。	

7 通用硬件加密模块

7.1 功能概述

加密模块自动完成指定长度的源地址空间数据的加密，完成后自动将加密数据回写到指定的目的地址空间；支持 SHA1/MD5/RC4/DES/3DES/AES/CRC/TRNG。

7.2 主要特征

- 支持 SHA1/MD5/RC4/DES/3DES/AES/CRC/TRNG 加密算法
- DES/3DES 支持 ECB 和 CBC 两种模式
- AES 支持 ECB、CBC 和 CTR 三种模式
- CRC 支持 CRC8、CRC16_MODBUS、CRC16_CCITT 和 CRC32 四种模式
- CRC 支持输入/输出反向
- SHA1/MD5/CRC 支持连续多包加密
- 内置真随机数发生器，也支持 seed 种子产生伪随机数

7.3 功能描述

7.3.1 SHA1 加密

可对连续多包数据进行硬件 SHA1 计算，计算结果存在寄存器中，上一包的加密结果可以作为下一包的初始值。

7.3.2 MD5 加密

可对连续多包数据进行硬件 MD5 计算，计算结果存在寄存器中，上一包的加密结果可以作为下一包的初始值。

7.3.3 RC4 加密

支持 RC4 加密和解密。

7.3.4 DES 加密

支持 DES 加密和解密，支持 ECB 和 CBC 两种模式。

7.3.5 3DES 加密

支持 3DES 加密和解密，支持 ECB 和 CBC 两种模式。

7.3.6 AES 加密

支持 AES 加密和解密，支持 ECB、CBC 和 CTR 三种模式。

7.3.7 CRC 加密

可对连续多包数据进行硬件 CRC 计算，计算结果存在寄存器中，上一包的加密结果可以作为下一包的初始值，支持 CRC8、CRC16_MODBUS、CRC16_CCITT 和 CRC32 四种模式，支持输入/输出反向。

CRC32 计算公式如下：

1、CRC-32: 0x04C11DB7

X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1

常用于 ZIP, RAR, IEEE 802 LAN/FDDI, IEEE 1394, PPP-FCS 等协议。

2、CRC-16：支持两种多项式

2.1: 0X1021

X16 + X12 + X5 + 1

常用于 ISO HDLC, ITU X.25, V.34/V.41/V.42, PPP-FCS CCITT 等协议。

2.2: 0X8005

X16 + X15 + X2 + 1

常用于 USB, ANSI X3.28, SIA DC-07 等协议。

3、CRC-8：0X207

x8+x2+x1+1

7.3.8 TRNG 随机数发生器

W800 系统中集成真随机数发生器模块。分为模拟模块与数字后处理模块。模拟模块输出随机时钟 ad_trng_clks 与随机数 ad_trng_dout， 数字后处理模块用来消除随机数的偏差与自相关性。相关控制寄存器在 GPSEC 寄存器列表中。

基本操作流程如下：

1. 使能 TRNG_EN，设置 TRNG_SEL 为 1，使得 GPSEC 寄存器 0x48 显示 TRNG 的输出值。此时模拟模块开始输出随机时钟及随机信号。前 8 个时钟采样得到的信号作为 LFSR 的初始状态初始化 LFSR 链，后面每个随机时钟采样得到的数据经过 XOR CHAIN 及 LFSR 寄存器的后处理，移位存储到寄存器 TRNG_RANDOM 中。
2. 软件可通过 GPSEC 寄存器 0x48 读取随机值。寄存器 TRNG_DIG_BYPASS 设置为 1 时数字后处理模块停止工作，直接将模拟模块的输出值存储到结果寄存器 TRNG_RANDOM 中。

7.4 寄存器描述

7.4.1 寄存器列表

表 31 加密模块寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	源地址寄存器	SRC_ADDR	RW	RC4/SHA1/AES/DES/3DES/CRC/MD5 复用源地址	0X0000_0000
0X0004	目的地址寄存器	DEST_ADDR	RW	RC4/AES/DES/3DES 复用目的地址	0X0000_0000
0X0008	配置寄存器	GPSEC_CFG	RW	通用硬件加密模块配置寄存器	0X0000_0000
0X000C	控制寄存器	GPSEC_CTRL	RW	通用硬件加密模块控制寄存器	0X0000_0000
0X0010	秘钥 0 低位寄存器	KEY00	RW	Key0 低 32 位第一个输入 key（RC4/AES/DES/3DES），复用 CRC Ci	0X0000_0000
0X0014	秘钥 0 高位寄存器	KEY01	RW	Key0 高 32 位第一个输入 key（RC4/AES/DES/3DES）	0X0000_0000
0X0018	秘钥 1 低位寄存器	KEY10	RW	Key1 低 32 位第二个输入 key（RC4/AES//3DES）	0X0000_0000
0X001C	秘钥 1 高位寄存器	KEY11	RW	Key1 高 32 位第二个输入 key（RC4/AES//3DES）	0X0000_0000
0X0020	秘钥 2 低位寄存器	KEY20	RW	Key2 低 32 位第三个输入 key（3DES），复用 iv1 低 32 位输入初始向量（AES）	0X0000_0000

0X0024	秘钥 2 高位寄存器	KEY21	RW	Key2 高 32 位第三个输入 key（3DES），复用 iv1 高 32 位输入初始向量（AES）	0X0000_0000
0X0028	初始向量 0 低位寄存器	IV00	RW	IV0 低 32 位输入初始向量（AES/DES/3DES）	0X0000_0000
0X002C	初始向量 0 高位寄存器	IV01	RW	IV0 高 32 位输入初始向量（AES/DES/3DES）	0X0000_0000
0X0030	状态寄存器	GPSEC_STS	RW	通用硬件加密模块状态寄存器	0X0000_0000
0X0034	摘要 0 寄存器	SHA1-DIGEST0	RW	sha1-digest0/MD5-digest0	0X6745_2301
0X0038	摘要 1 寄存器	SHA1-DIGEST1	RW	sha1-digest1/MD5-digest1	0XEFCD_AB89
0X003C	摘要 2 寄存器	SHA1-DIGEST2	RW	sha1-digest2/MD5-digest2	0X98BA_DCFE
0X0040	摘要 3 寄存器	SHA1-DIGEST3	RW	sha1-digest3/MD5-digest3	0X1032_5476
0X0044	摘要 4 寄存器	SHA1-DIGEST4	RW	sha1-digest4/CRC	0XC3D2_E1F0
0X0048	RNG_result	RNG_RESULT	RW	RNG 输出	0X0000_0000
0X004C	秘钥 3 低位寄存器	Key30	RW	Key3 低 32 位第三个输入 key（RC4 的 256bit 模式）	0X0000_0000
0X0050	秘钥 3 低位寄存器	Key31	RW	Key3 高 32 位第三个输入 key（RC4 的 256bit 模式）	0X0000_0000
0X0054	TRNG 配置	TRNG_CR	RW	真随机数发生器配置选择	0X40

7.4.2 配置寄存器

表 32 加密模块配置寄存器

位	访问	操作说明	复位值
[31]	RW	RC4_128_256 0：标志 RC4 加密/解密是按照 128bit 块长度进行； 1：标志 RC4 加密/解密是按照 256bit 块长度进行。	1'b0
[30]	RW	RNG start 1'b0：不启动 RNG 1'b1：启动 RNG	1'b0
[29]	RW	RNG Load_seed 硬件自动清 0 1'b0：随机数发生器会默认以零为种子，产生相应位数的随机数 1'b1：种子加载完成以后开始生成随机数	1'b0
[28]	RW	RNG switch 控制生成随机数的位数， 1'b0：16 位 1'b1：32 位	1'b0
[27]	RW	des_soft_reset des 软复位完成后硬件自动清 0 1'b0：不产生软复位且不改变当前状态 1'b1：加密算法被软件复位为初始状态	1'b0
[26]	RW	aes_soft_reset aes 软复位完成后硬件自动清 0 1'b0：不产生软复位且不改变当前状态	1'b0

		1'b1：加密算法被软件复位为初始状态	
[25]	RW	rc4_soft_reset rc4 软复位完成后硬件自动清 0 1'b0：不产生软复位且不改变当前状态 1'b1：加密算法被软件复位为初始状态	1'b0
[24]	RW	crc_datarev 1'b0：CRC 输入数据不反向 1'b1：CRC 输入数据反向	1'b0
[23]	RW	crc_chksrev 1'b0：CRC 输出结果不反向 1'b1：CRC 输出结果反向	1'b0
[22:21]	RW	sub_mode 算法类型子模式选择： 0：DES/AES 密码算法的 ECB 模式，可复用 CRC 算法的 CRC8 模式 1：DES/AES 密码算法的 CBC，可复用 CRC 算法的 CRC16_0 模式 2：AES 密码算法的 CTR 模式，可复用 CRC 算法的 CRC16_1 模式 3： AES 密码算法的 MAC 模式，可复用 CRC 算法的 CRC32	2'b0
[20]	RW	encrypt_decrypt RC4/AES/DES/3DES 算法的加密或解密模式选择： 1'b0：加密 1'b1：解密	1'b0
[19]	RW	gpsec_int_mask	1'b0

		1’b0：不屏蔽加/解密完成中断 1’b1：屏蔽加/解密完成中断	
[18:16]	RW	cypher_mode 密码算法类型 3’b000：RSV 3’b001：RC4 3’b010：SHA1 3’b011：AES 3’b100：DES 3’b101：3DES 3’b110：CRC 3’b111：MD5	3’b0
[15: 0]	RW	total_byte 总共需要加解密操作的 byte 个数。	16’h0

7.4.3 TRNG 控制寄存器

表 33 TRNG 模块控制寄存器

位	访问	操作说明	复位值
[31: 7]		保留	
[6]	RW	TRNG_INT_MASK TRNG 中断 mask	1’b1

		0：TRNG 模块上报中断； 1：TRNG 模块不上报中断；	
[5:3]	RW	TRNG_CP TRNG 模块控制信号	3’d0
[2]	RW	TRNG_DIG_BYPASS TRNG 数字后处理模块 bypass： 0：TRNG 模块进行后处理； 1：TRNG 模块不进行后处理；	1’b0
[1]	RW	TRNG_SEL: RNG 输出选择信号。 0：寄存器 0x48 显示 伪随机模块输出结果； 1：寄存器 0x48 显示 TRNG 输出结果；	1’b0
[0]	RW	TRNG_EN: TRNG 模块使能信号。高有效。 0：TRNG 模块停止； 1：TRNG 模块开始工作；	1’b0

7.4.4 控制寄存器

表 34 加密模块控制寄存器

位	访问	操作说明	复位值
[31: 2]		保留	

[1]	RW	sec_stop 停止当前正在进行的加解密操作 1'b0：无效 1'b1：加/解密停止	1'b0
[0]	RW	sec_strt 启动加解密，完成加解密操作的 byte 个数后，硬件自动清 0 1'b0：不启动加/解密 1'b1：启动加/解密	1'b0

7.4.5 状态寄存器

表 35 加密模块状态寄存器

位	访问	操作说明	复位值
[31: 17]		保留	
[16]	RW	int_flag 软件写 1 清零 1'b0：不产生加/解密完成中断 1'b1：产生加/解密完成中断	1'b0
[15: 0]	RO	transfer_cnt 当前加密完成的字节个数。 每次重新开启加解密时清 0，并重新开始计数。	16'h0

8 RSA 加密模块

8.1 功能概述

RSA 运算硬件协处理器，提供 Montgomery(FIOS 算法)模乘运算功能。配合 RSA 软件库实现 RSA 算法。支持 128 位到 2048 位模乘。

8.2 主要特征

- 支持 128 位到 2048 位模乘（模乘长度是 32 位的整数倍）
- 支持 D*D；X*Y；D*Y；X*X 等 4 个模乘模式

8.3 功能描述

8.3.1 模乘功能

RSA 运算硬件协处理器，提供 Montgomery（FIOS 算法）模乘运算功能。配合 RSA 软件库一同实现 RSA 算法。支持 128 位到 2048 位模乘。

8.4 寄存器描述

8.4.1 寄存器列表

表 36 RSA 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000~0X00FC	数据 X 寄存器	XBUF	RW	数据 X 寄存器	
0X0100~0X01FC	数据 Y 寄存器	YBUF	RW	数据 Y 寄存器	
0X0200~0X02FC	数据 M 寄存器	MBUF	RW	数据 M 寄存器	
0X0300~0X03FC	数据 D 寄存器	DBUF	RW	数据 D 寄存器	

0X0400	RSA 控制寄存器	RSACON	RW	RSA 控制寄存器	0X0000_0000
0X0404	参数 MC 寄存器	RSAMC	WO	参数 MC 寄存器	0X0000_0000
0X0408	参数 N 寄存器	RSAN	RW	参数 N 寄存器	0X0000_0000

8.4.2 数据 X 寄存器

XBUF 对应数据 X（2048bit）的缓冲区，对应 haddr 值为 0000h~00fch。对应规则如下表：

表 37 RSA 数据 X 寄存器

000h	004h	008h	.....	00f8h	00fch
X[31:0]	X[63:32]	X[95:64]	.....	X[2015:1984]	X[2047:2016]

8.4.3 数据 Y 寄存器

YBUF 对应数据 Y（2048bit）的缓冲区，对应 haddr 值为 0100h~01fch。对应规则如下表：

表 38 RSA 数据 Y 寄存器

0100h	0104h	0108h	.....	01f8h	01fch
Y[31:0]	Y[63:32]	Y[95:64]	.....	Y[2015:1984]	Y[2047:2016]

8.4.4 数据 M 寄存器

MBUF 对应数据 M（2048bit）的缓冲区，对应 haddr 值为 0200h~02fch。对应规则如下表：

表 39 RSA 数据 M 寄存器

0200h	0204h	0208h	.....	02f8h	02fch
M[31:0]	M[63:32]	M[95:64]	.....	M[2015:1984]	M[2047:2016]

8.4.5 数据 D 寄存器

DBUF 对应数据 D（2048bit）的缓冲区，对应 haddr 值为 0300h~03fch。对应规则如下表：

表 40 RSA 数据 D 寄存器

0300h	0304h	0308h	.....	03f8h	03fch
D[31:0]	D[63:32]	D[95:64]	.....	D[2015:1984]	D[2047:2016]

8.4.6 RSA 控制寄存器

RSACON，RSA 控制寄存器，实际物理空间为 32bit 寄存器。

表 41 RSA 控制寄存器

位	访问	操作说明	复位值
[31: 6]		保留	
[5]	RW	模乘启动控制位。软件写“1”启动模乘运算，运算结束后，硬件自动清“0”。	1’b0
[4]	RW	提供软复位功能，高有效。软件写“1”进行软复位，复位完成后，硬件自动清“0”。 1.设置参数 MC 和 N 为 0。 2.启动模乘后（bit5 置 1），将此位置“1”，会终止当前的运算（当 bit0 变高，表示软复位命令执行完成，运算被终止），但内部数据缓冲区（X，Y，M，D）中已经完成的部分运算结果会保留。	1’b0
[2 : 3]	RW	模乘模式选择。 2’b00：X = D*D mod M 2’b01：D = X*Y mod M 2’b10：X = D*Y mod M 2’b11：D = X*X mod M	2’b0
[1]	RW	保留	1’b0

[0]	RW	模乘运算完成标识，高有效。硬件置“1”，软件清“0”。软件写“1”无效。	1’b0

8.4.7 参数 MC 寄存器

表 42 RSA 参数 MC 寄存器

位	访问	操作说明	复位值
[31: 0]	WO	RSAMC 对应参数 MC（32bit）。复位值全 0。读出值为全 0。	32’h0

8.4.8 参数 N 寄存器

RSAN 对应参数 N（7bit）。N 值为模乘长度除以 32 的值。即如果调用 1024bit 的模乘运算，需设置 N = 32。对该寄存器写入时，取低 7 位为有效数据，读出时，高 25 位为 0。复位值全 0。

表 43 RSA 参数 N 寄存器

位	访问	操作说明	复位值
[31: 7]		保留	
[6 : 0]	RW	RSAN 对应参数 N（7bit）。N 值为模乘长度除以 32 的值。	7’h0

9 GPIO 模块

9.1 功能概述

GPIO 控制器实现了软件对 GPIO 属性的配置，使用户能够方便的操作 GPIO。

每个 GPIO 都可以通过软件单独配置，设置其作为输入端口、输出端口，设置其悬浮、上拉、下拉状态，设置其上升沿、下降沿、双沿、高电平、低电平中断触发方式。

9.2 主要特性

- 支持 GPIO 软件配置
- 支持 GPIO 中断配置
- 最多提供 48 个 GPIO 可用

9.3 功能描述

W800 中提供的 GPIO 分为两组，一组为 GPIOA，一组为 GPIOB，GPIOA 和 GPIOB 寄存器起始地址不同，但是功能一致。

当用户希望将某个特定 IO 作为软件控制的 GPIO 使用的话，将 GPIO 复用选择寄存器中对应位置为 0 即可。

GPIO 方向控制寄存器用来控制 GPIO 的方向，1 表示对应的 GPIO 作为输出引脚，0 表示对应的 GPIO 作为输入引脚。

GPIO 上下拉控制寄存器用来控制相应 IO 的上下拉功能。

GPIO 上拉控制寄存器为低有效，设置为 0 表示打开相应 IO 的上拉功能，设置为 1 表示关闭上下拉功能。

IO 具有的属性请参见 IO 复用表。

GPIO 下拉控制寄存器为高有效,设置为 1 表示打开相应 IO 的下拉功能,设置为 0 表示关闭上下拉功能。

IO 具有的属性请参见 IO 复用表。

GPIO 数据寄存器在设置为输入状态时表示输入 IO 的电平，在设置为输出状态时可以写入 1 或者 0 指定 IO 的输出电平。此寄存器受到 GPIO 数据使能寄存器的控制，只有在 GPIO 数据使能寄存器设置为 1 的时候，GPIO 数据寄存器才能读写。

GPIO 模块提供输入信号检测功能。通过配置 GPIO 中断相关的寄存器可以实现高低电平检测以及上下沿跳变检测。当对应 IO 的输入信号符合预先设置的条件，比如说高电平触发或者上升沿触发等，即会触发 GPIO 中断，上报给 MCU 处理。MCU 需要清除相应的中断状态，以免中断误触发。

9.4 寄存器描述

9.4.1 寄存器列表

表 44 GPIOA 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	GPIO 数据寄存器	GPIO_DATA	RW	读写 GPIO 当前数据	0X180B
0X0004	GPIO 数据使能寄存器	GPIO_DATA_EN	RW	配置 GPIO_DATA 的使能位	0XFFFF
0X0008	GPIO 方向控制寄存器	GPIO_DIR	RW	配置 GPIO 方向	0X0000
0X000C	GPIO 上拉控制寄存器	GPIO_PULL_EN	RW	配置 GPIO 上拉	0XFFFF
0X0010	GPIO 复用选择寄存器	GPIO_AF_SEL	RW	配置 GPIO 复用功能使能位	0XFFFF

0X0014	GPIO 复用选择寄存器 1	GPIO_SF_S1	RW	GPIO 复用功能选择位高地址位	0X0000
0X0018	GPIO 复用选择寄存器 0	GPIO_AF_S0	RW	GPIO 复用功能选择位低地址位	0X0000
0X001C	GPIO 下拉控制寄存器	GPIO_DN_ENA	RW	配置 GPIO 下拉	0X0000
0X0020	GPIO 中断触发方式配置寄存器	GPIO_IS	RW	配置 GPIO 的中断触发方式	0X0000
0X0024	GPIO 中断边沿触发模式配置寄存器	GPIO_IBE	RW	配置 GPIO 中断边沿触发模式	0X0000
0X0028	GPIO 中断上下边沿触发配置寄存器	GPIO_IEV	RW	配置 GPIO 中断上下边沿触发或高低电平触发	0X0000
0X002C	GPIO 中断使能配置寄存器	GPIO_IE	RW	配置 GPIO 中断使能	0X0000
0X0030	GPIO 裸中断状态寄存器	GPIO_RIS	RO	查询 GPIO 裸中断状态（MASK 前）	0X0000
0X0034	GPIO 屏蔽后中断状态寄存器	GPIO_MIS	RO	查询 GPIO 屏蔽后中断状态（MASK 后）	0X0000
0X0038	GPIO 中断清除控制寄存器	GPIO_IC	WO	控制 GPIO 中断清除	0X0000

表 45 GPIOB 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	GPIO 数据寄存器	GPIO_DATA	RW	读写 GPIO 当前数据	0X0000_7304
0X0004	GPIO 数据使能寄存器	GPIO_DATA_EN	RW	配置 GPIO_DATA 的使能位	0X7FFF_FFFF
0X0008	GPIO 方向控制寄存器	GPIO_DIR	RW	配置 GPIO 方向	0X0000_0000
0X000C	GPIO 上拉控制寄存器	GPIO_PULL_EN	RW	配置 GPIO 上拉	0XFFFF_FFFF

0X0010	GPIO 复用选择寄存器	GPIO_AF_SEL	RW	配置 GPIO 复用功能使能位	0XFFFF_FFFF
0X0014	GPIO 复用选择寄存器 1	GPIO_SF_S1	RW	GPIO 复用功能选择位高地址位	0X0000_0000
0X0018	GPIO 复用选择寄存器 0	GPIO_AF_S0	RW	GPIO 复用功能选择位低地址位	0X0000_0000
0X001C	GPIO 下拉控制寄存器	GPIO_DN_ENA	RW	配置 GPIO 下拉	0X0000_0000
0X0020	GPIO 中断触发方式配置寄存器	GPIO_IS	RW	配置 GPIO 的中断触发方式	0X0000_0000
0X0024	GPIO 中断边沿触发模式配置寄存器	GPIO_IBE	RW	配置 GPIO 中断边沿触发模式	0X0000_0000
0X0028	GPIO 中断上下边沿触发配置寄存器	GPIO_IEV	RW	配置 GPIO 中断上下边沿触发或高低电平触发	0X0000_0000
0X002C	GPIO 中断使能配置寄存器	GPIO_IE	RW	配置 GPIO 中断使能	0X0000_0000
0X0030	GPIO 裸中断状态寄存器	GPIO_RIS	RO	查询 GPIO 裸中断状态（MASK 前）	0X0000_0000
0X0034	GPIO 屏蔽后中断状态寄存器	GPIO_MIS	RO	查询 GPIO 屏蔽后中断状态（MASK 后）	0X0000_0000
0X0038	GPIO 中断清除控制寄存器	GPIO_IC	WO	控制 GPIO 中断清除	0X0000_0000

9.4.2 GPIO 数据寄存器

表 46 GPIOA 数据寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 当前数据，每 BIT 与相应的 GPIO 线对应	16’h180b

表 47 GPIOB 数据寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 当前数据，每 BIT 与相应的 GPIO 线对应	32'h7304

9.4.3 GPIO 数据使能寄存器

表 48 GPIOA 数据使能寄存器

位	访问	操作说明	复位值
[15: 0]	RW	对应 GPIO_DATA 的 BIT 使能位，只有对应 BIT 为 1 时，对 GPIO_DATA 相应位的操作才有效，每 BIT 与相应的 GPIO 线对应，1'bx： [x] = 0, GPIO_DATA[x]不可读写 [x] = 1, GPIO_DATA[x]可读写	16'hffff

表 49 GPIOB 数据使能寄存器

位	访问	操作说明	复位值
[31: 0]	RW	对应 GPIO_DATA 的 BIT 使能位，只有对应 BIT 为 1 时，对 GPIO_DATA 相应位的操作才有效，每 BIT 与相应的 GPIO 线对应，1'bx： [x] = 0, GPIO_DATA[x]不可读写 [x] = 1, GPIO_DATA[x]可读写	32'h7fff_ffff

9.4.4 GPIO 方向控制寄存器

表 50 GPIOA 方向控制寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 方向控制，每 BIT 与相应的 GPIO 线对应，1'bx：	16'h0

		[x] = 0, GPIO[x]为输入 [x] = 1, GPIO[x]为输出	

表 51 GPIOB 方向控制寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 方向控制，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]为输入 [x] = 1, GPIO[x]为输出	32'h0

9.4.5 GPIO 上下拉控制寄存器

表 52 GPIOA 上拉控制寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 上拉控制，每 BIT 与相应的 GPIO 线对应，1'bx: 注意：该寄存器为低有效 [x] = 0, GPIO[x]有上拉 [x] = 1, GPIO[x]无上拉	16'hffff

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 下拉控制，每 BIT 与相应的 GPIO 线对应，1'bx: 注意：该寄存器为高有效 [x] = 1, GPIO[x]有下拉 [x] = 0, GPIO[x]无下拉	16'h0000

表 53 GPIOB 上下拉控制寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 上拉控制，每 BIT 与相应的 GPIO 线对应，1'bx： 注意：该寄存器为低有效 [x] = 0, GPIO[x]有上拉 [x] = 1, GPIO[x]无上拉	32'hffff_ffff

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 下拉控制，每 BIT 与相应的 GPIO 线对应，1'bx： 注意：该寄存器为高有效 [x] = 1, GPIO[x]有下拉 [x] = 0, GPIO[x]无下拉	32'h0000_0000

9.4.6 GPIO 复用选择寄存器

表 54 GPIOA 复用选择寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 复用功能使能位，每 BIT 对应相应 GPIO 复用功能是否打开，1'bx：	16'hffff

		[x] = 0, GPIO[x]复用功能关闭 [x] = 1, GPIO[x]复用功能打开 [x] = 1 时，复用功能取决于 GPIO_AF_S1 和 GPIO_AF_S0 两个寄存器相应 BIT 的状态。 S1.[x] = 0， S0.[x] = 0，复用功能 1(opt1) S1.[x] = 0， S0.[x] = 1，复用功能 2(opt2) S1.[x] = 1， S0.[x] = 0，复用功能 3(opt3) S1.[x] = 1， S0.[x] = 1，复用功能 4(opt4) [x] = 0 时，如果 GPIO_DIR[x] = 0，且 GPIO_PULL_EN[x] = 1，则 GPIO 复用为 opt6 模拟 IO 功能 IO 复用功能参见芯片引脚复用关系	

表 55 GPIOB 复用选择寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 复用功能使能位，每 BIT 对应相应 GPIO 复用功能是否打开，1'bx： [x] = 0, GPIO[x]复用功能关闭 [x] = 1, GPIO[x]复用功能打开 [x] = 1 时，复用功能取决于 GPIO_AF_S1 和 GPIO_AF_S0 两个寄存器相应 BIT 的状态。 S1.[x] = 0， S0.[x] = 0，复用功能 1(opt1)	32'hffff_ffff

		S1.[x] = 0， S0.[x] = 1，复用功能 2(opt2) S1.[x] = 1， S0.[x] = 0，复用功能 3(opt3) S1.[x] = 1， S0.[x] = 1，复用功能 4(opt4) [x] = 0 时，如果 GPIO_DIR[x] = 0，且 GPIO_PULL_EN[x] = 1，则 GPIO 复用为 opt6 模拟 IO 功能 IO 复用功能参见芯片引脚复用关系	

9.4.7 GPIO 复用选择寄存器 1

表 56 GPIOA 复用选择寄存器 1

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 复用功能选择位高地址位，和 GPIO_AF_S0 共同决定复用功能 IO 复用功能参见芯片引脚复用关系	16’h0

表 57 GPIOB 复用选择寄存器 1

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 复用功能选择位高地址位，和 GPIO_AF_S0 共同决定复用功能 IO 复用功能参见芯片引脚复用关系	32’h0

9.4.8 GPIO 复用选择寄存器 0

表 58 GPIOA 复用选择寄存器 0

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 复用功能选择位低地址位，和 GPIO_AF_S1 共同决定复用功能 如何配置参见 GPIO_AF_SEL 寄存器说明	16'h0

表 59 GPIOB 复用选择寄存器 0

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 复用功能选择位低地址位，和 GPIO_AF_S1 共同决定复用功能 如何配置参见 GPIO_AF_SEL 寄存器说明	32'h0

9.4.9 GPIO 中断触发方式配置寄存器

表 60 GPIOA 中断触发方式配置寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 的中断触发方式，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]中断为边沿触发 [x] = 1, GPIO[x]中断为电平触发	16'h0

表 61 GPIOB 中断触发方式配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 的中断触发方式，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]中断为边沿触发	32'h0

		[x] = 1, GPIO[x]中断为电平触发	

9.4.10 GPIO 中断边沿触发模式配置寄存器

表 62 GPIOA 中断边沿触发模式配置寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 中断边沿触发模式，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]边沿触发中断模式由 GPIO_IEV 决定 [x] = 1, GPIO[x]双沿都触发中断	16'h0

表 63 GPIOB 中断边沿触发模式配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 中断边沿触发模式，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]边沿触发中断模式由 GPIO_IEV 决定 [x] = 1, GPIO[x]双沿都触发中断	32'h0

9.4.11 GPIO 中断上下边沿触发配置寄存器

表 64 GPIOA 中断上下边沿触发配置寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 中断上下边沿触发或高低电平触发选择，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]中断为低电平或下降沿触发 [x] = 1, GPIO[x]中断为高电平或上升沿触发	16'h0

表 65 GPIOB 中断上下边沿触发配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 中断上下边沿触发或高低电平触发选择，每 BIT 与相应的 GPIO 线对应， 1'bx： [x] = 0, GPIO[x]中断为低电平或下降沿触发 [x] = 1, GPIO[x]中断为高电平或上升沿触发	32'h0

9.4.12 GPIO 中断使能配置寄存器

表 66 GPIOA 中断使能配置寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 中断使能控制，每 BIT 与相应的 GPIO 线对应，1'bx： [x] = 0, GPIO[x]中断失能 [x] = 1, GPIO[x]中断使能	16'h0

表 67 GPIOB 中断使能配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 中断使能控制，每 BIT 与相应的 GPIO 线对应，1'bx： [x] = 0, GPIO[x]中断失能 [x] = 1, GPIO[x]中断使能	32'h0

9.4.13 GPIO 裸中断状态寄存器

表 68 GPIOA 裸中断状态寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 裸中断状态（MASK 前），每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]没有中断产生 [x] = 1, GPIO[x]有中断产生	16'h0

表 69 GPIOB 裸中断状态寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 裸中断状态（MASK 前），每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]没有中断产生 [x] = 1, GPIO[x]有中断产生	32'h0

9.4.14 GPIO 屏蔽后中断状态寄存器

表 70 GPIOA 屏蔽后中断状态寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 屏蔽后中断状态（MASK 后），每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]没有中断产生（MASK 后） [x] = 1, GPIO[x]中断产生（MASK 后）	16'h0

表 71 GPIOB 屏蔽后中断状态寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 屏蔽后中断状态（MASK 后），每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0, GPIO[x]没有中断产生（MASK 后） [x] = 1, GPIO[x]中断产生（MASK 后）	32'h0

9.4.15 GPIO 中断清除控制寄存器

表 72 GPIOA 中断清除控制寄存器

位	访问	操作说明	复位值
[15: 0]	RW	GPIO 中断清除控制，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0，无动作 [x] = 1，清除 GPIO[x]中断状态	16'h0

表 73 GPIOB 中断清除控制寄存器

位	访问	操作说明	复位值
[31: 0]	RW	GPIO 中断清除控制，每 BIT 与相应的 GPIO 线对应，1'bx: [x] = 0，无动作 [x] = 1，清除 GPIO[x]中断状态	32'h0

10 高速 SPI 设备控制器

10.1功能概述

兼容通用 SPI 物理层协议，通过约定与主机交互的数据格式，主机可以对设备进行高速访问，最高支持工作频率为 50MHZ。

10.2主要特性

- 兼容通用 SPI 协议
- 可选择的电平中断信号
- 最高支持 50Mbps 速率
- 简单的帧格式，全硬件解析与 DMA

10.3 功能描述

10.3.1 SPI 协议简介

SPI 以主从方式工作，通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以（单向传输时）。分别是 SDI（数据输入）、SDO（数据输出）、SCLK（时钟）、CS（片选）。

（1）SDI - Serial Data In，串行数据输入

（2）SDO - Serial Data Out，串行数据输出

（3）SCLK - Serial Clock，时钟信号，由主设备产生

（4）CS - Chip Select，从设备使能信号，由主设备控制。

其中，CS 是从芯片是否被主芯片选中的控制信号，也就是说只有片选信号为预先规定的使能信号时（高

电位或低电位），主芯片对此从芯片的操作才有效。这就使在同一条总线上连接多个 SPI 设备成为可能。

除了上述 4 根信号线之后，HSPI 还额外增加了一根 INT 线，当从设备有数据需要上传时，产生一个下降沿的中断，实现数据的主动上报。

SPI 通讯是通过数据交换完成的，数据是一位一位的传输的，由 SCLK 提供时钟脉冲，SDI，SDO 则基于此脉冲完成数据传输。数据输出通过 SDO 线，数据在时钟上升沿或下降沿时改变，在紧接着的下降沿或上升沿被读取。完成一位数据传输，输入也使用同样原理。因此，至少需要 8 次时钟信号的改变（上沿和下沿为一次），才能完成 8 位数据的传输。

SCLK 信号线由主设备控制，从设备不能控制信号线。在一个基于 SPI 的设备中，至少有一个主控设备。

10.3.2 SPI 工作过程

芯片内部的 HSPI 是和 wrapper 控制器一起工作的，wrapper 控制器内部集成 DMA，通过 DMA 实现 HSPI 内部 FIFO 和芯片内部缓存之间的数据交换。该操作是硬件实现的，软件不需要关心数据发送接收过程，只需要配置发送接收数据链表，以及操作 wrapper 控制器相应的寄存器。

关于 wrapper 控制器的详细介绍，请参考相关章节。

10.4寄存器描述

10.4.1 HSPI 芯片内部操作的寄存器列表

表 74 HSPI 内部访问寄存器

偏移地址	名称	缩写	访问	描述	复位值

0X0000	HSPI FIFO 清空寄存器	CLEAR_FIFO	RW	清除 Tx 和 Rx FIFO 的内容，同时会同步复位系统时钟域的电路	0X0000_0000
0X0004	HSPI 配置寄存器	SPI_CFG	RW	配置 SPI 的传输模式以及大小端配置	0X0000_0000
0X0008	HSPI 模式配置寄存器	MODE_CFG	RW	配置 ahb master 访问总线时的 burst 长度	0X0000_0000
0X000C	HSPI 中断配置寄存器	SPI_INT_CPU_MASK	RW	配置中断是否使能	0X0000_0003
0X0010	HSPI 中断状态寄存器	SPI_INT_CPU_STTS	RW	获取以及清除中断状态	0X0000_0000
0X0018	HSPI 数据上传长度寄存器	RX_DAT_LEN	RW	配置可以上传的数据长度	0X0000_0000

10.4.1.1 HSPI FIFO 清空寄存器

表 75 HSPI FIFO 清空寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	Clear FIFOs，清除 Tx 和 Rx FIFO 的内容，同时会同步复位系统时钟域的电路（本列表中的寄存器除外） 0：不清除 FIFO 1：清除有效 软件置位，硬件清零 注：如果要复位整个电路，需要采用本模块的异步复位管腿：rst_n	1'b0

10.4.1.2 HSPI 配置寄存器

表 76 HSPI 配置寄存器

位	访问	操作说明	复位值
[31: 4]	RO	保留	
[3]	RW	Bigendian, spi 接口支持数据的大小端选择。 0：支持小段数据传输 1：支持大端数据传输	1'b0
[2]	RW	spi_tx_always_drive 0：spi 输出只有在片选有效时有效，其它时刻为高阻 1：spi 输出一直有效	1'b0
[1]	RW	SPI CPHA 0：传输模式 A 1：传输模式 B	1'b0
[0]	RW	SPI CPOL，SCK 在 IDLE 时的极性 0：SCK IDLE 时为 0 1：SCK IDLE 时为 1	1'b0

10.4.1.3 HSPI 模式配置寄存器

表 77 HSPI 模式配置寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	

[0]	RW	Burst len，ahb master 访问总线时的 burst 长度 0：burst len 为 1 字 1：burst len 为 4 字 建议设置为 4 个字的 burst 传输，这样在 spi 接口频率较高时，可以保证不断流	1'b0

10.4.1.4 HSPI 中断配置寄存器

表 78 HSPI 中断配置寄存器

位	访问	操作说明	复位值
[31: 2]	RO	保留	
[1]	RW	IntEnRxOverrun，RxOverrun 中断使能 0：Rx FIFO overflow 中断使能 1：Rx FIFO overflow 中断不使	1'b1
[0]	RW	IntEnTxUnderrun，TxUnderrun 中断使能 0：Tx FIFO underflow 中断使能 1：Tx FIFO underflow 中断不使能	1'b1

10.4.1.5 HSPI 中断状态寄存器

表 79 HSPI 中断状态寄存器

位	访问	操作说明	复位值
[31: 2]	RO	保留	
[1]	RW	RxOverrun 0：Rx FIFO overflow	1'b0

		1：Rx FIFO overflow 写 1 清零	
[0]	RW	TxUnderrun 0：Tx FIFO　underflow 1：Tx FIFO　underflow 写 1 清零	1'b0

10.4.1.6 HSPI 数据上传长度寄存器

表 80 HSPI 数据上传长度寄存器

位	访问	操作说明	复位值
[31:16]	RO	保留	
[15: 0]	RW	Rx_dat_len 表示可以上传的数据长度，单位为字节 上传长度都是字的整数倍，如果上传长度不足整字，向上取整。	16'h0

10.4.2 主机端访问 HSPI 控制器寄存器列表

主机端通过固定的 SPI 命令格式访问 SPI 接口寄存器。命令长度固定为一个字节，数据长度固定为两个字节。

表 81 HSPI 接口配置寄存器(主设备访问)

偏移地址	名称	缩写	访问	描述	复位值

0X02	获取数据长度寄存器	RX_DAT_LEN	RO	上传数据时，spi 主机用来获取从 device 端读取的数据长度	0X0000
0X03	下发数据标志寄存器	TX_BUFF_AVAIL	RO	主向从发数据时，用于判断是否可以下传数据或者命令	0X0000
0X04	保留	RSV	RO		
0X05	中断配置寄存器	SPI_INT_HOST_MASK	RW	是否屏蔽中断	0X0000
0X06	中断状态寄存器	SPI_INT_HOST_STTS	RO	中断状态寄存器，spi 主机查询该位确认是否有数据上传	0X0000
0X07	保留	RSV	RO		
0X00	数据端口 0	DAT_PORT0	RW	Spi 主机通过该寄存器端口向从设备下发数据，下发前面的数据帧使用该端口	
0X10	数据端口 1	DAT_PORT1	RW	Spi 主机通过该寄存器端口向从设备下发数据，下发最后一个数据帧使用该端口	
0X01	命令端口 0	DN_CMD_PORT0	WO	Spi 主机通过该寄存器向从设备下发命令数据，下发前面的命令数据使用该端口	
0X11	命令端口 1	DN_CMD_PORT1	WO	Spi 主机通过该寄存器向从设备下发命令数据，下发最后一帧命令数据使用该端口	

10.4.2.1 HSPI 获取数据长度寄存器

表 82 HSPI 获取数据长度寄存器

位	访问	操作说明	复位值
[15: 0]	RO	spi 主机只读寄存器，在上传数据时，主要用来获知从 device 端读取多少数据 但在本模块中，上传长度都是字的整数倍，如果此上传长度值不为整字，则主机读数时向上取整，即多读部分冗余字节	16'h0

10.4.2.2 HSPI 下发数据标志寄存器

表 83 HSPI 下发数据标志寄存器

位	访问	操作说明	复位值
[15: 2]	RO	保留	
[1]	RO	tx_cmdbuff_avail 标志发送 cmd 的 buff 是否可用，如果可用，则主机可以下发 cmd。 0：发送 buff 不可用 1：发送 buff 可用	1'b0
[0]	RO	tx_buff_avail 标志发送 buff 是否可用，如果可用，则主机可以下发数据。 0：发送 buff 不可用 1：发送 buff 可用	1'b0

10.4.2.3 HSPI 中断配置寄存器

表 84 HSPI 中断配置寄存器

位	访问	操作说明	复位值
[15: 1]	RO	保留	
[0]	RO	IntMaskup_dat_cmd_rdy 中断屏蔽 0：中断没有被屏蔽，可以产生中断 1：中断被屏蔽 注： 建议采用主机自己内部的中断屏蔽，这样可以提高效率。	1'b0

10.4.2.4 HSPI 中断状态寄存器

表 85 HSPI 中断状态寄存器

位	访问	操作说明	复位值
[15: 1]	RO	保留	
[0]	RO	up_dat_cmd_rdy 向 SPI 主机产生中断的状态寄存器 0：数据或命令没有准备好 1：数据或命令已经准备好 读可清	1'b0

10.4.2.5 HSPI 数据端口 0

表 86 HSPI 数据端口 0

位	访问	操作说明	复位值
	RW	SPI 主机通过该寄存器端口和 device 进行数据传输，向该寄存器写数，即可下发数据，从该寄存器读数，即可上传数据。如果正在传输的帧需要通过多次传输才能完成，则最后一次传输采用寄存器端口 DAT_PORT1，其它的采用 DAT_PORT0。	

10.4.2.6 HSPI 数据端口 1

表 87 HSPI 数据端口 1

位	访问	操作说明	复位值
	RW	SPI 主机通过该寄存器端口和 device 进行数据传输，向该寄存器写数，即可下发数据，从该寄存器读数，即可上传数据。如果正在传输的帧需要通过多次传输才能完成，则最后一次传输采用寄存器端口 DAT_PORT1，其它的采用 DAT_PORT0。	

10.4.2.7 HSPI 命令端口 0

表 88 HSPI 命令端口 0

位	访问	操作说明	复位值
	RW	SPI 主机通过该寄存器端口和 device 进行命令交互，向该寄存器写数，即可下发命令。如果正在传输的命令需要通过多次传输才能完成，则最后一次传输采用寄存器端口 DN_CMD_PORT1，其它的采用 DN_CMD_PORT0。 注：此窗口只用来下发驱动和固件协商的命令。	

10.4.2.8 HSPI 命令端口 1

表 89 HSPI 命令端口 1

位	访问	操作说明	复位值
	RW	SPI 主机通过该寄存器端口和 device 进行命令交互，向该寄存器写数，即可下发命令。如果正在传输的命令需要通过多次传输才能完成，则最后一次传输采用寄存器端口 DN_CMD_PORT1，其它的采用 DN_CMD_PORT0。 注：此窗口只用来下发驱动和固件协商的命令。	

10.4.3 高速 SPI 设备控制器接口时序

主要描述 SPI 读写时序，以及主 SPI 如何与 HSPI 进行数据交互。

10.4.3.1 数据格式

数据格式分为命令域和数据域两部分，如下图。其中命令域固定长度为 8bit，数据域长度根据访问对象不同，长度不同，具体参见下文。

命令域的最高 bit 为读写标志位，其余 7bit 为地址.

- 0 表示从后边 7bit 地址处读数据
- 1 表示向后边 7bit 地址写数据

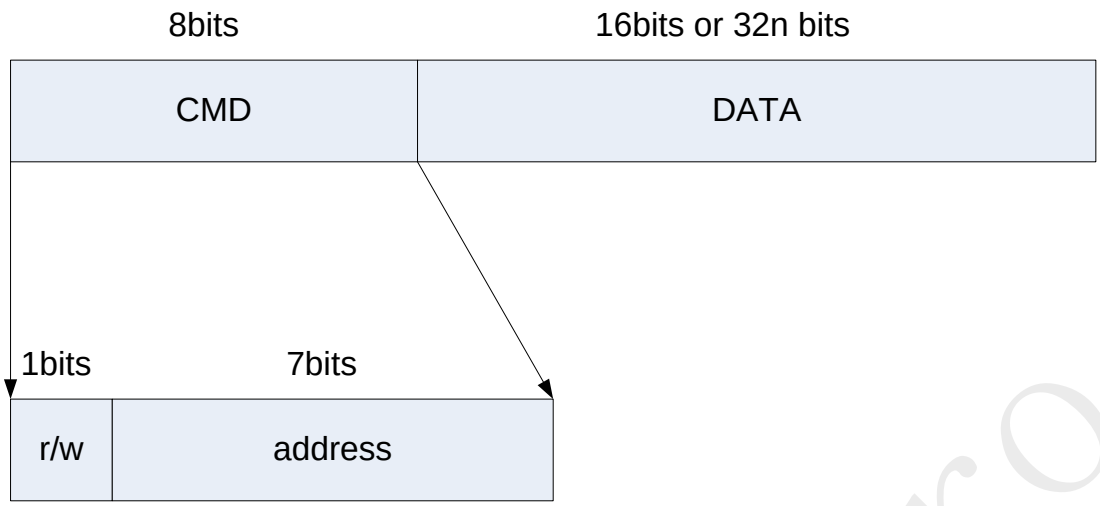


图 5 上位机 SPI 收发数据格式

本模块数据域只支持两种长度，上位机 SPI 访问接口配置寄存器（表 2），数据域长度为 16bit；

通过端口（数据端口 0，数据端口 1，命令端口 0 和命令端口 1）传输数据，数据域长度为 32bit 的整数倍；

下图为读写接口配置寄存器的时序图。从设备默认配置是小端模式。

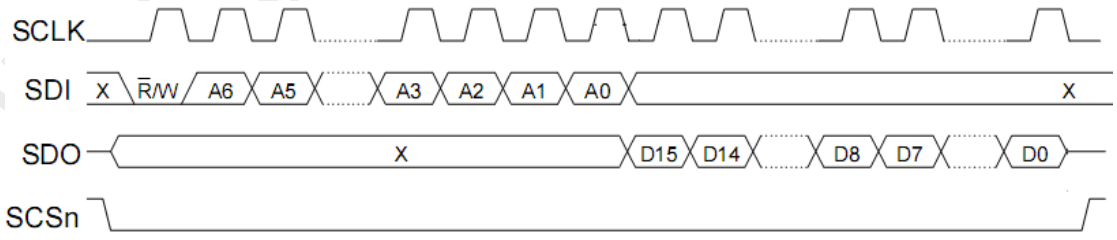


图 6 HSPI 寄存器读操作(大端模式)

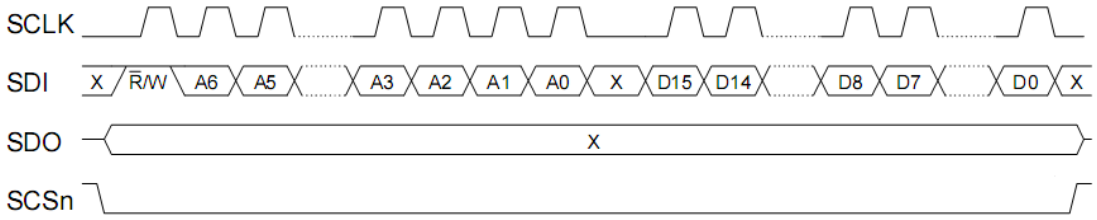


图 7 HSPI 寄存器写操作(大端模式)

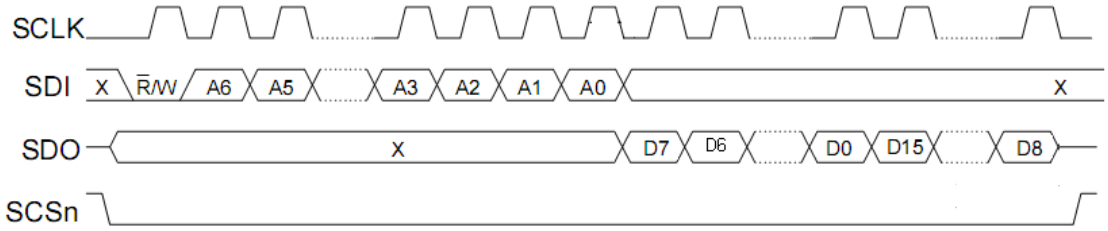


图 8 寄存器读操作(小端模式)

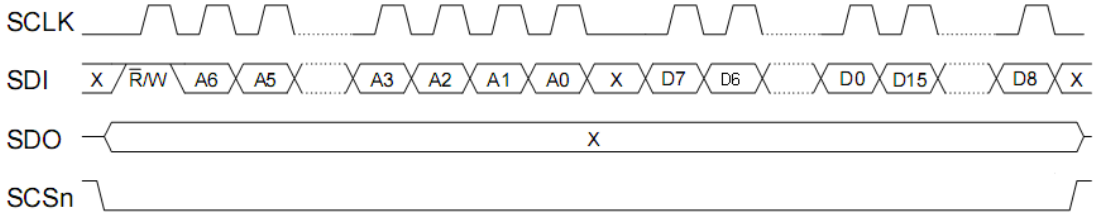


图 9 寄存器写操作(小端模式)

下图为读写数据的时序图，数据域长度为 32bit 的整数倍，图示只传输一个字的长度。

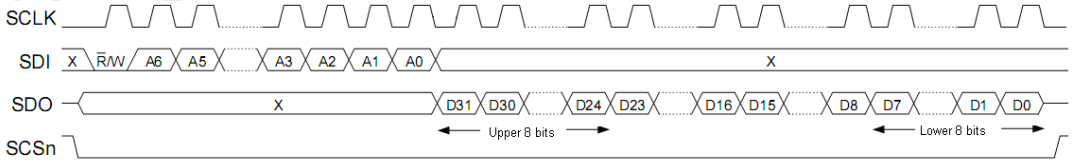


图 10 端口读操作(大端模式)

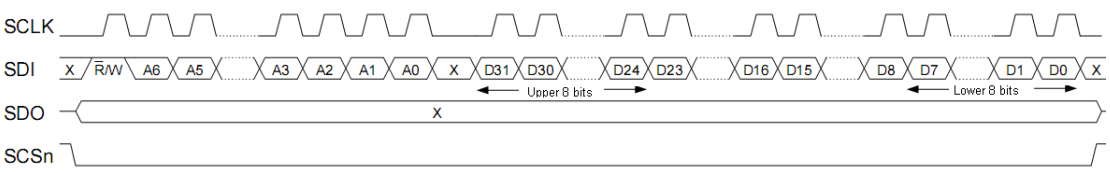


图 11 端口写操作(大端模式)

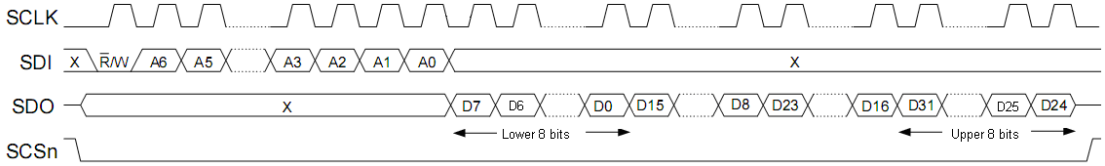


图 12 端口读操作(小端模式)

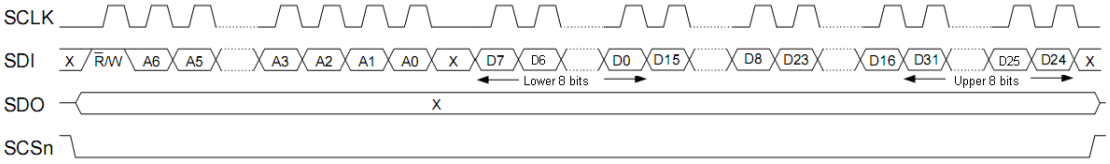


图 13 端口写操作(小端模式)

注：命令和数据之间可以没有等待时间，即传输命令字段后，可以紧接着数据传输，不需要多余空闲时钟或者空闲时间。有时间延迟也可以，但不能出现空闲时钟。

10.4.3.2 时序

本模块支持半双工，可以支持的时序根据时钟相位和采样点的不同，分为 4 种。以下时序只是给出时钟的相位和采样关系。需要注意的是，芯片默认支持（CPOL=0,CPHA=0）。

注：命令和数据之间可以没有等待时间，即传输命令字段后，可以紧接着数据传输，不需要多余空闲时钟或者空闲时间。有时间延迟也可以，但不能出现空闲时钟。

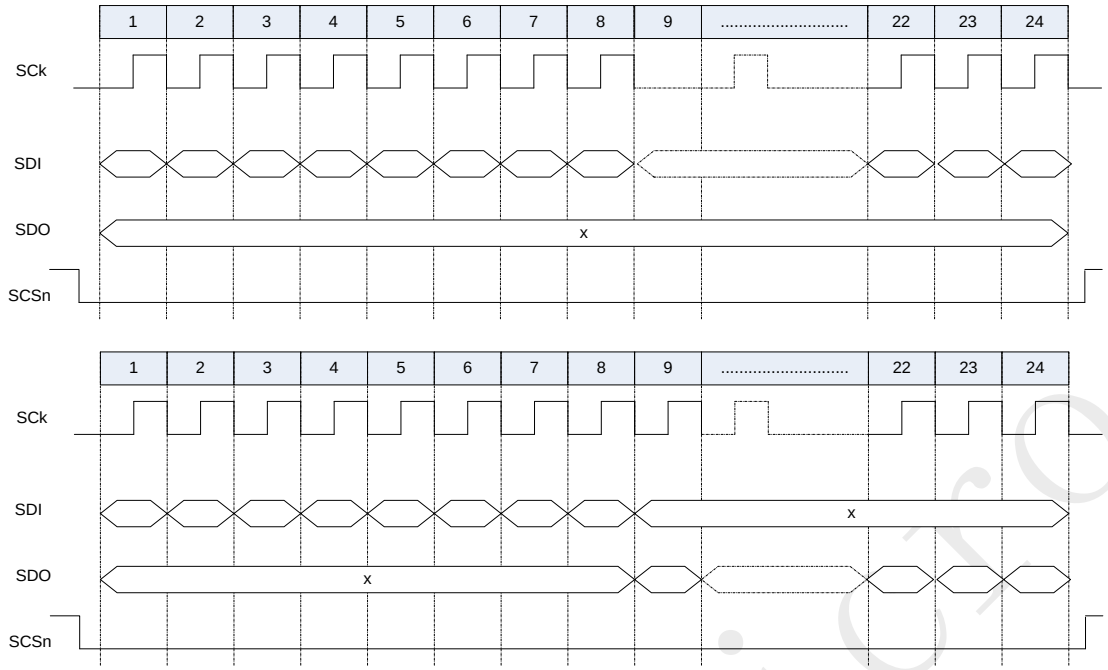


图 14 CPOL=0,CPHA=0

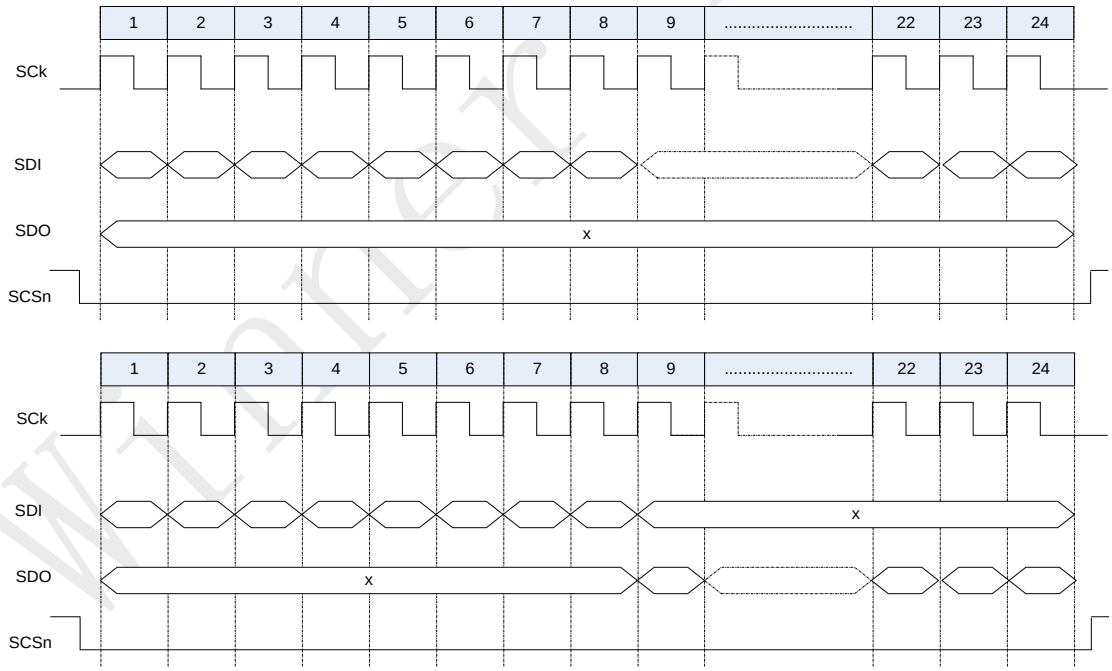


图 15 CPOL=0,CPHA=1

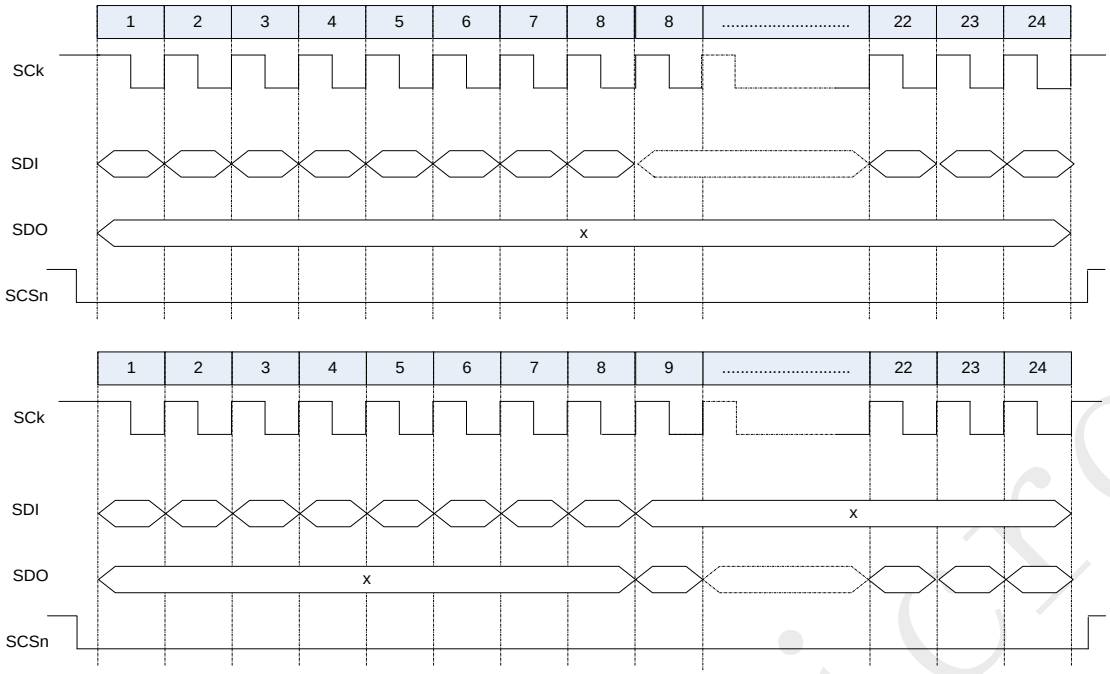


图 16 CPOL=1,CPHA=0

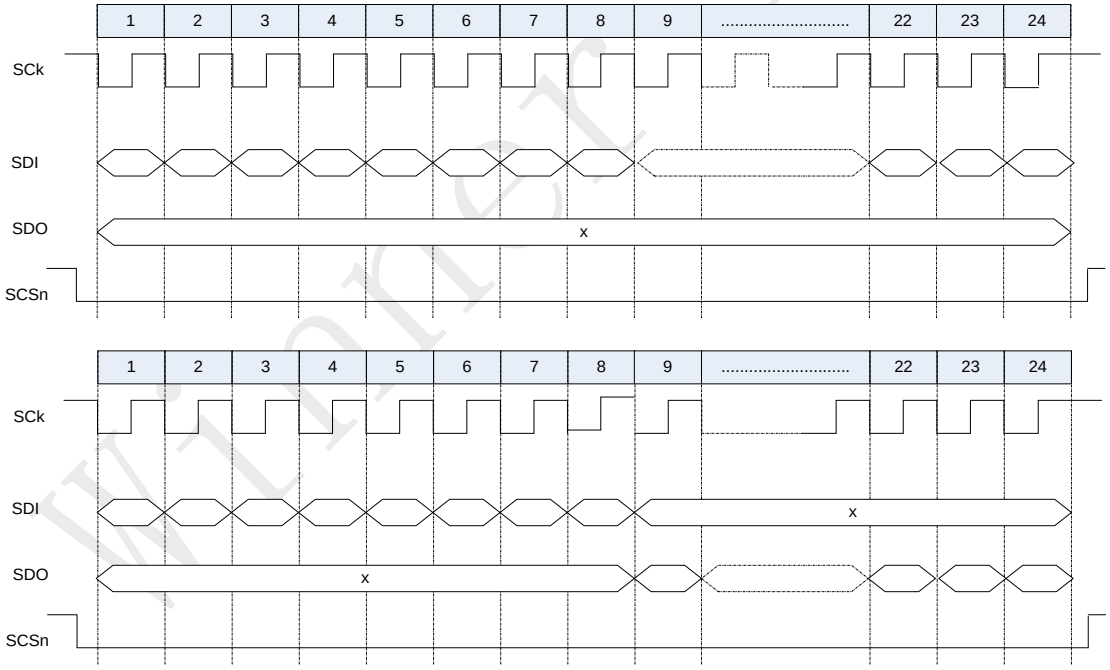


图 17 CPOL=1,CPHA=1

10.4.3.3 中断

中断信号由从设备发送给主设备，通过 SPI_INT 管脚触发，低电平有效。

spi_int 主要通知 spi 主机有数据或者命令需要上传，spi 主机处理中断时关心的接口寄存器为：

- SPI_INT_HOST_MASK
- SPI_INT_HOST_STTS
- RX_DAT_LEN

注：每一上传的帧对应一个中断。只有当前需要上传的帧传输完成后，如果还有帧需要上传，此时，才会产生新的中断。下图是处理中断的一种方式。

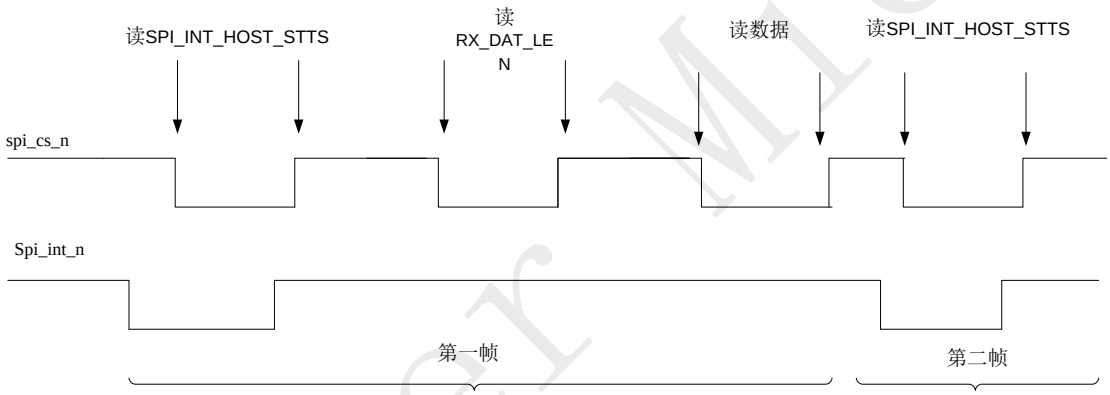


图 18 主 SPI 处理中断流程

10.4.3.4 主 SPI 收发数据工作流程

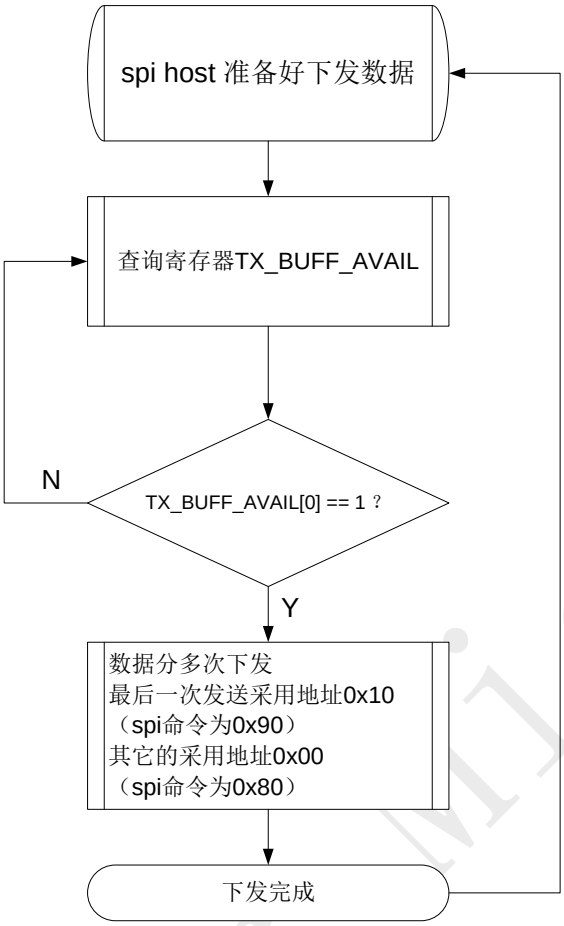


图 19 下行数据流程图

注意：下发的数据的长度必须是以字为单位，如果非整字，填 0 补齐。

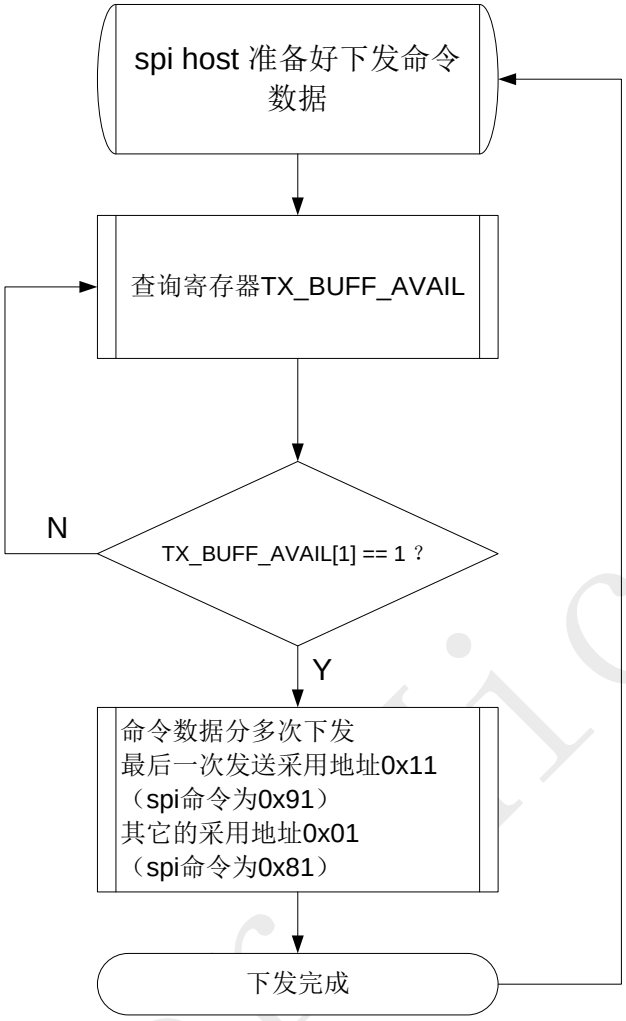


图 20 下行命令流程图

注意：下发的命令的长度必须是以字为单位，如果非整字，填 0 补齐。

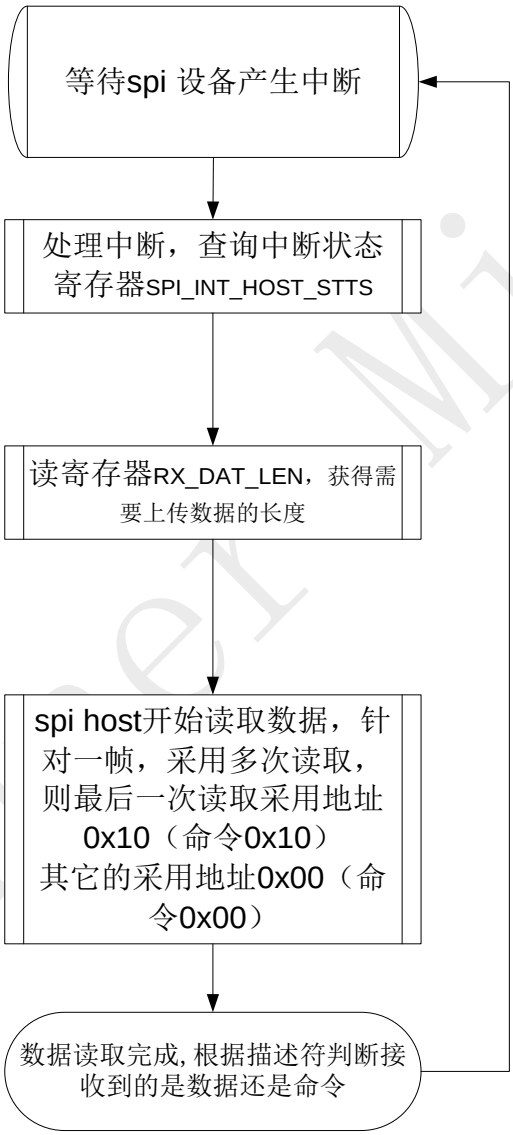


图 21 上行数据(命令)流程图

上行数据和上行命令的流程一样。

此处需要注意的是，上行的数据的长度必须是以字为单位，如果有效长度非整字，尾部多余数据可以扔

掉。

需要说明的是，主从之间有数据和命令两个通道可以交互数据，用户可以根据需要任意选择一个通道使用或者两个都使用。命令通道一次交互最大数据长度为 256 字节，数据通道一次交互最大数据长度为 1500 字节。数据长度限制由从设备控制，如果长度超限，会破坏从设备的数据结构。

11 SDIO 设备控制器

11.1 功能概述

W800 集成了 SDIO 设备端接口，作为从设备，完成与主机数据的交互。内部集成了 1024byte 的异步 FIFO，完成主机与芯片的数据交互。

11.2 主要特性

- 兼容 SDIO 卡规范 2.0
- 支持主机速率 0~50MHz
- 支持最大 1024 字节的 Block
- 支持 1 比特 SD 和 4 比特 SD 模式

11.3 功能描述

11.3.1 SDIO 总线

SDIO 总线和 USB 总线类似，SDIO 总线也有两端，其中一端是主机端，另一端是设备端，采用 HOST-DEVICE 这样的设计是为了简化 DEVICE 的设计，所有的通信都是由 HOST 端发出命令开始的。在 DEVICE 端只要能解析 HOST 的命令，就可以同 HOST 进行通信了，SDIO 的 HOST 可以连接多个 DEVICE。

在 SDIO 总线定义中,DAT1 信号线复用为中断线。在 SDIO 的 1BIT 模式下 DAT0 用来传输数据，DAT1 用作中断线。在 SDIO 的 4BIT 模式下 DAT0-DAT3 用来传输数据，其中 DAT1 复用作中断线。

11.3.2 SDIO 命令

SDIO 总线上都是 HOST 端发起请求，然后 DEVICE 端回应请求，其中请求和回应中会包含数据信息：

- Command:用于开始传输的命令，是由 HOST 端发往 DEVICE 端的，其中命令是通过 CMD 信号线传送的；
- Response:回应是 DEVICE 返回的,作为 Command 的回应。也是通过 CMD 线传送的；
- Data:数据是双向的传送的。可以设置为 1 线模式，也可以设置为 4 线模式。数据是通过 DAT0-DAT3 信号线传输的。

SDIO 的每次操作都是由 HOST 在 CMD 线上发起一个 CMD，对于有的 CMD，DEVICE 需要返回 Response，有的则不需要。

对于读命令，首先 HOST 会向 DEVICE 发送命令，紧接着 DEVICE 会返回一个握手信号，此时，当 HOST 收到回应的握手信号后，会将数据放在 4 位的数据线上，在传送数据的同时会跟随着 CRC 校验码。当整个读传送完毕后，HOST 会再次发送一个命令，通知 DEVICE 操作完毕，DEVICE 同时会返回一个响应。

对于写命令，首先 HOST 会向 DEVICE 发送命令，紧接着 DEVICE 会返回一个握手信号，此时，当 HOST 收到回应的握手信号后，会将数据放在 4 位的数据线上，在传送数据的同时会跟随着 CRC 校验码。当整个写传送完毕后，HOST 会再次发送一个命令，通知 DEVICE 操作完毕，DEVICE 同时会返回一个响应。

11.3.3 SDIO 内部存储

SDIO 设备内部具有固定的存储映射，包括一般资讯区域（CIA）和特殊功能区域（function unique area）。

CIA 中的寄存器包括了 I/O 端口功能，中断产生以及端口工作信息，可以通过读写功能 0 对 CIA 所定义的寄存器进行相关操作。CIA 包含了 CCCR，FBR 和 CIS 共三方面信息。其中 CCCR 定义了 SDIO 卡的公

共控制寄存器,主机端通过操作 CCCR 可以对 SDIO 卡进行检查和对端口进行操作,CCCR 的地址为 0X00-0XFF。FBR 定义了所支持的端口功能 1 到端口功能 7 的操作,包括了各端口的要求和功能,电源控制等,FBR 的地址为 0Xn00-0Xnff（其中 n 为功能端口号)。CIS 定义了卡的一些信息结构，地址为 0X1000-0X17FFF，CIS 有公共 CIS 和各功能端口各自的 CIS，其中公共 CIS 的初始地址在 CCCR 的 CIS Pointer 域中，各端口功能的 CIS 在各功能端口 FBR 的 CIS Pointer 域中。

CIA 的存储映射如下图。

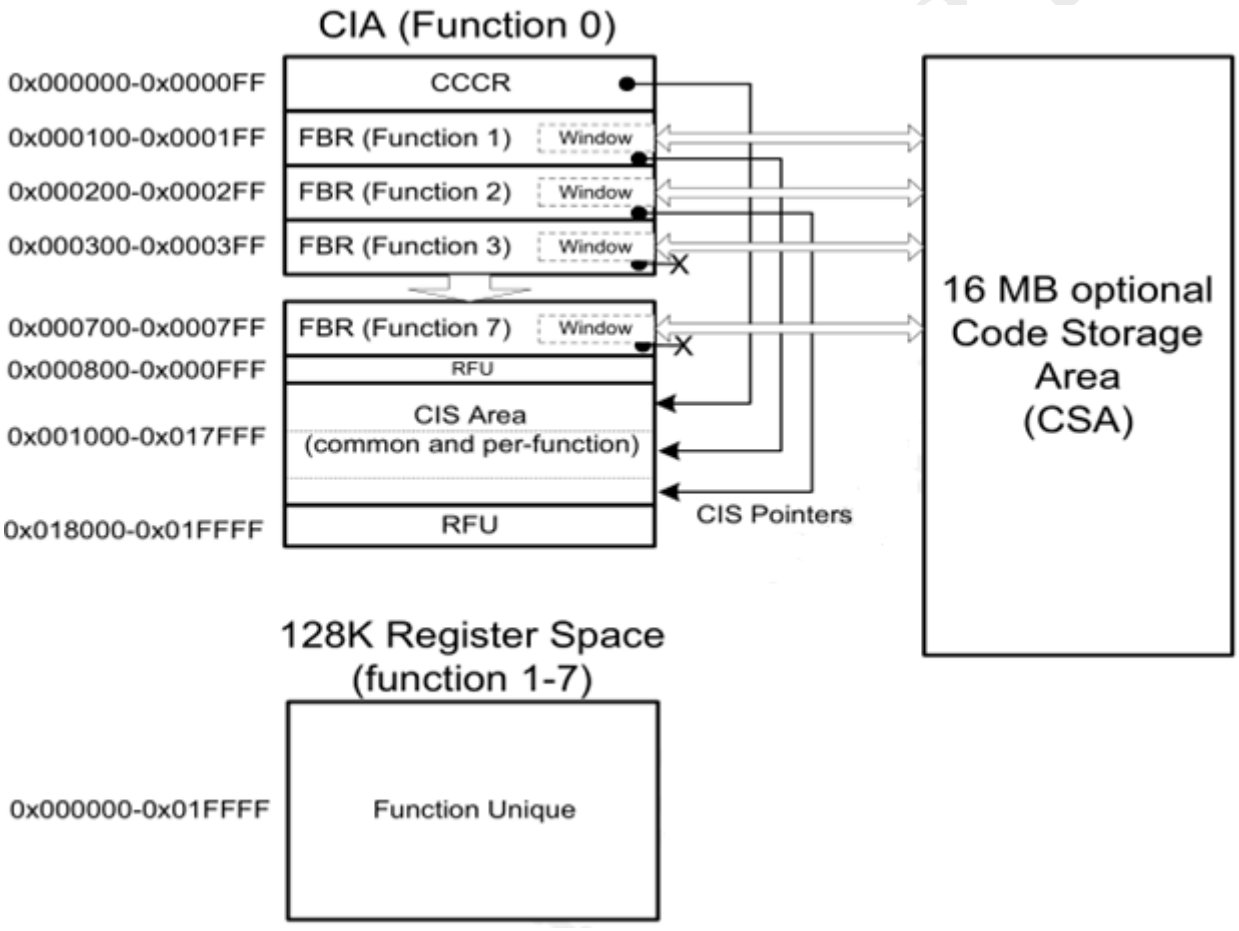


图 22 SDIO 内部存储映射

CIA 中各寄存器的描述参考下文。要深入了解 CIA，请参阅 SDIO 协议规范。

11.4 寄存器描述

11.4.1 寄存器列表

11.4.2 SDIO Fn0 寄存器

Fn0 寄存器为 SDIO 协议规定的寄存器，其地址范围为：0x00000~0x1FFFF，共 128K。起始地址为 0x00000。

Fn0 寄存器由 SDIO 主机通过 CMD52 命令进行访问，偏移地址即为访问地址，功能号为 0。

Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	CCCR/SDIO Revision	SDIO bit 3	SDIO bit 2	SDIO bit 1	SDIO bit 0	CCCR bit 3	CCCR bit 2	CCCR bit 1	CCCR bit 0
0x01	SD Specification Revision	RFU	RFU	RFU	RFU	SD bit 3	SD bit 2	SD bit 1	SD bit 0
0x02	I/O Enable	IOE7	IOE6	IOE5	IOE4	IOE3	IOE2	IOE1	RFU
0x03	I/O Ready	IOR7	IOR6	IOR5	IOR4	IOR3	IOR2	IOR1	RFU
0x04	Int Enable	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IENM
0x05	Int Pending	INT7	INT6	INT5	INT4	INT3	INT2	INT1	RFU
0x06	I/O Abort	RFU	RFU	RFU	RFU	RES	AS2	AS1	AS0
0x07	Bus Interface Control	CD Disable	SCSI	ECSI	RFU	RFU	RFU	Bus Width 1	Bus Width 0
0x08	Card Capability	4BLS	LSC	E4MI	S4MI	SBS	SRW	SMB	SDC
0x09-0x0B	Common CIS Pointer	Pointer to card's *common* Card Information Structure (CIS)							
0x0C	Bus Suspend	RFU	RFU	RFU	RFU	RFU	RFU	BR	BS
0x 0D	Function Select	DF	RFU	RFU	RFU	FS3	FS2	FS1	FS0
0x 0E	Exec Flags	EX7	EX6	EX5	EX4	EX3	EX2	EX1	EXM
0x 0F	Ready Flags	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RFM
0x10-0x11	FN0 Block Size	I/O block size for Function 0							
0x 12	Power Control	Reserved for Future Use (RFU)						EMPC	SMPC
0x 13	High-Speed	RFU	RFU	RFU	RFU	RFU	RFU	EHS	SHS
0x14-0xEF	RFU	Reserved for Future Use (RFU)							
0xF0-0xFF	Reserved for Vendors	Area Reserved for Vendor Unique Registers							

图 23 CCCR 寄存器存储结构

Address	7	6	5	4	3	2	1	0
0x100	Function 1 CSA enable	Function 1 supports CSA	RFU	RFU	Function 1 Standard SDIO Function interface code			
0x101	Function 1 Extended standard SDIO Function interface code							
0x102	RFU	RFU	RFU	RFU	RFU	RFU	EPS	SPS
0x103-0x108	Reserved for Future Use (RFU)							
0x109-0x10B	Pointer to Function 1 Card Information Structure (CIS)							
0x10C-0x10E	Pointer to Function 1 Code Storage Area (CSA)							
0x10F	Data access window to Function 1 Code Storage Area (CSA)							
0x110-0x111	I/O block size for Function 1							
0x112-0x1FF	Reserved for Future Use							
0x200-0x7FF	Function 2 to 7 Function Basic Information Registers (FBR)							
0x800-0xFFF	Reserved for Future Use							

图 24 FBR1 寄存器结构

Address	7	6	5	4	3	2	1	0
0x0001000 - 0x017FFF	Card Common Card Information Structure (CIS) area for card common and all functions							
0x018000- 0x01FFFF	Reserved for Future Use							

图 25 CIS 存储空间结构

11.4.2.1 SDIO CCCR 寄存器和 FBR1 寄存器列表

表 90 SDIO CCCR 寄存器和 FBR1 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X00	CCCR/SDIO Revision	SDIOx	RO	[3:0],表示支持的 CCCR/FBR 格式 4'h0： CCCR/FBR Version 1.00 4'h1：CCCR/FBR Version 1.10 4'h2：CCCR/FBR Version 1.20 4'h3-4'hF：Rsv 由 CIA Register[3:0]来表示	4'h2

		CCCRx	RO	[7:4],表示支持的 SDIO 协议版本 4’h0： SDIO Version 1.00 4’h1： SDIO Version 1.10 4’h2： SDIO Version 1.20 (unreleased) 4’h3： SDIO Version 2.00 4’h4-4’hF： Rsv 由 CIA Register[7:4]来表示	4’h3
0X01	SD pecificatio n Revision	SDx	RO	[3:0]，表示支持的 SD 协议版本 4’h0：SD Physical Version 1.01 (March 2000) 4’h1：SD Physical Version 1.10 (October 2004) 4’h2：SD Physical Version 2.00 (May 2006) 4’h3-4’hF： Rsv 由 CIA Register[11:8]来表示	4’h2
			RO	RFU	4’h0
0X02	I/O Enable	IOEx	RO	RFU	1’b0
			RW	[7:1]，Function 使能，bit1-bit7 分别对应 7 个 function，对应 SD 主机使相应 bit 为 1，则使能相应 function，否则相应 function 不能工作。 注：CIS0，CIS1 以及 CSA 放在 Fn1 中，此时即使 Fn1 不使能，SD 主机也可以对这个三个区域进行读写（CIS0，CIS1 不可以写）。	7’b0
0X03	I/O Ready	IORx	RO	RFU	1’b0

			RO	[7:1],IOR 共 7bit，分别对应 7 个 function 的状态，如果对应 bit 为 1，则表示该 function 可以工作。 本设计中，HC8051 通过配置 program register 的 function ready 位为 1，使的本寄存器 bit1=1，从而标志 Fn1 可以正常工作。 注：对 CIS0、CIS1、CSA 的读写操作独立于 IOR1，即，即使 IOR1=0，也可以访问这三个存储空间的内容。	7'b0
0X04	Int Enable	IENM	RW	[0],中断使能信号 0：来至卡里的中断不能送给 SD 主机 1：任何 function 的中断都可以送给主机	1'b0
		IENx	RW	[7:1],functionx 的中断使能。 IEN1=0,则来至 Fn1 的中断不会送给主机。 IEN1=1，则允许 Fn1 的中断送给主机	7'b0
0X05	Int Pending		RO	[0],RFU	1'b0
		INTx	RO	[7:1],functionx 的中断挂起。 INT1=0，没有 Fn1 的中断被挂起 INT1=1，Fn1 有中断被挂起。 注：如果 IEN1 和 IENM 不为 1，则主机不会收到挂起中断	7'b0
0X06	I/O Abort	ASx	WO	[2:0],取消 IO 读或者写，从而释放总线。 取消 Fn1 操作，应该采用 CMD52 写写入 3'b1。该命令在 SPI 下不支持。	3'b0
		RES	WO	[3],软复位信号	1'b0

				1：复位 SD 时钟域的电路，该 bit 在置位后自动清零，不需要专门清零。该复位信号不会影响当前卡的协议选择（SD 或者 SPI 模式），同时也不会影响 CD Disable。只能采用 CMD52 操作。	
			RO	RFU	4‘b0
0X07	Bus Interface ontrol	Bus Width	RW	[1:0]，数据线宽度 2’b00：1bit 数据线模式 2’b10：4bit 数据线模式 复位或者上电，会变为 2’b00	2’b00
			RO	[4:2],RFU	3’b000
		ECSI	RW	[5],允许连续的 SPI 中断。 如果 SCSI 为 1，则这一个寄存器用来允许 SDIO 卡在 SPI 模式下，在任何时间给出中断，此时不需要关心 CS 线的状态。	1’b0
		SCSI	RO	[6]，支持连续的 SPI 中断。 如果为 1，表示 SDIO 卡支持 SPI 模式下，在任何时候给出中断，而不需要关心 CS 的状态。 当 progtam reg[2]为 1，该寄存器置位。	1’b1
		CD Disable	RW	[7]，连接或者断开 CD/DAT[3](pin1)上的 10-90K 上拉电阻。 0：连接上拉电阻 1：断开上拉电阻 在上电后，该寄存器清零，即连接上拉电阻。该寄存器的状态不会被 SD 协议中的复位命令影响。	1’b0
0X08	Card	SDC	RO	[0],支持在数据传输期间执行 CMD52 命令。在 SPI 模式下不支持	1’b1

	Capability			该寄存器。 当 progtam reg[3]为 1 时，该寄存器为 1	
		SMB	RO	[1],表示 SDIO 卡支持 CMD53 要求的 Block 传输模式。 当 program_reg[4]为 1 时，该寄存器为 1	1'b1
		SRW	RO	[2],表示 SDIO 支持读等待-Read Wait Control（RWC）操作。 当 program_reg[5]为 1 时，该寄存器为 1	1'b1
		SBS	RO	[3] ,表示 SDIO 卡支持挂起/恢复。 如果为 0，则不支持(0x0C-0x0F)寄存器 如果为 1，除了 Fn0，所有 function 都会根据 SD 主机要求挂起或者恢复 当 program_reg[6]为 1 时，该寄存器为 1	1'b1
		S4MI	RO	[4],表示 SDIO 卡支持在 4bit 多 Block 数据传输模式下，向主机产生中断。 0：不支持在 Block 传输间产生中断，在这种情况下，只要 IENx=1，SDIO 还是可以在其他中断周期向主机发起中断 1：支持在 Block 传输间产生中断 当 program_reg[7]为 1 时，该寄存器为 1	1'b1
		E4MI	RW	[5],中断使能。 允许 4bit 多 Block 模式下，在两 block 数据传输中间向主机产生中断。 0：不允许 1：允许	1'b0

				上电复位或者复位命令会使该寄存器清 0	
		LSC	RO	[6], 0：表示 SDIO 卡为全速设备 1：表示 SDIO 卡为低速设备 当 program_reg[8]为 1 时，该寄存器为 1	1'b0
		4BLS	RO	[7], 0：表示 SDIO 为低速模式设备或者不支持 4bit 模式 1：表示 SDIO 为低速模式设备，且支持 4bit 模式 当 program_reg[9]为 1 时，该寄存器为 1	1'b1
0X09- 0X0B	Common CIS pointer		RO	[23:0],指向 SDIO 卡共用 CIS（CIS0）的起始地址指针。CIS0 包含关于整个卡的信息。其访问空间为：0x001000-0x017FFF。 指针以小端格式存储（LSB）。	24'h00 1000
0X0C	Bus Suspend	BS	RO	[0] ,总线状态。 0：当前选中的 function 没有使用数据总线 1：当前选中的 function（用 FSx 或者用 IO 命令中的 function number）正在执行将要在数据线上传输数据的命令 这个寄存器被主机用来决定哪个 function 当前正在使用数据总线。 如果 SDIO 卡不支持挂起恢复功能，则该寄存器为 0. 任何访问 CIA 的操作不能挂起，该寄存器一直为 1，即使 BR 寄存器为 1。	1'b0

				SPI 模式下，只读，且为 0.	
		BR	RW	总线释放 请求/状态。该寄存器用来请求选定的 function（用 FSx 或者 CMD53 总 function number 选中）释放数据总线并挂起相关操作。 如果主机设置该寄存器为 1，则选定的 function 将暂时停止在数据线上的数据传输，并挂起当前数据操作的命令。BR 寄存器保持为 1，直到释放过程完成。一旦 function 被挂起，设备通过清零 BS、BR 来通知主机。主机可以通过读 BR 来监测挂起请求的执行状态，如果 BR 为 1，则挂起请求仍在执行。主机可以主动向 BR 写 0 来取消正在执行的挂起请求。 SPI 模式下，只读，且为 0.	4’h0
			RO	[7:2],RFU	6’b0
0X0D	Function Select	FSx	RW	[3:0]，在挂起/恢复操作中，用来选取 function[0-7]。两种方法写 FSx： 对 CCCR 执行 IO 写操作 新发起的 IO 命令将导致 FSx 设置为命令中的 function number。 如果 function 当前被挂起，向 FSx 写该 function 的 number，则在读 FSx 时，将恢复该 function 的数据传输操作。返回的值将会是当前选中的 function 的 number。 注：当读 FSx 时，如果 BS=0，则 FSx 的值未定义。	4’b0

				4’b0000： Transaction of function 0 (CIA) 4’b0001-4’b0111： Transaction to functions 1-7 4’b1000： Transaction of memory in combo card 4’b1001-4’b1111： Not defined, reserved for future use	
			RO	[3:1],RFU	3’b000
		DF	RO	[7]，恢复数据标志。向 FSx 写入 function number，将恢复选中 function 的数据传输。一旦数据传输恢复，DF 寄存器将标明是否有更多的数据要传输。 0：在 function 被恢复后，没有更多数据要传输。 1：在 function 被恢复后，有更多数据要传输。 DF 用来在 4bit 模式下，控制中断周期。如果为 1，在 function 恢复后，有更多数据要传输，在这种情况下，中断周期被取消。如果为 0，function 在数据传输结束后恢复（在 busy 情况下），在这种情况下，在恢复后，没有数据传输，因此主机可以在 function 恢复后监测到中断周期的开始。	1’b0
0X0E	Exec Flags	EXx	RO	[7:0]，执行标志位。主机通过这些 bit 位来决定所有 function[7-1]执行命令的状态。这些寄存器可以告知主机某一 function 正在执行命令，因此不能对该 function 发出新的命令。 SPI 模式下，只读，且为 0.	8’h00
0X0F	Ready Flags	RFx	RO	[7:0]，读标志位。主机通过这些寄存器可以得知对 function[7-1]读写忙状态。如果一个 function 正在执行一个写交易，对应的	8’h00

				RFx bit 清零标志着该 function 忙，没有准备好接收更多的数据。如果一个 function 正在执行一个读操作，对应的 RFx bit 清零，则标明读数据无效，如果为 1，则标明读数据可以传输。 SPI 无效，只读，且为 0	
0X10- 0X11	FN0 Block Size		RW	[15:0],Fn0 对应的 Block 传输时，Block size 大小。最大 2048Byte，最小 1Byte。存储方式为小段格式（LSB）	16'h00
0X12	Power Control	SMPC	RO	[0],支持主机功耗控制。 0：SDIO 总电流小于 200mA，即使所有 function 都有效（IOEx=1）。EMPC、SPS、EPS 都为 0。 1：SDIO 总电流可以超过 200mA。EMPC、SPS、EPS 有效。	1'b1
		EMPC	RO	[1],主机功耗控制使能。 0：SDIO 卡总电流小于 200mA。SDIO 卡自动切换 function(s) 到低电流模式或者不允许一些 function 使能，而其忽略 EPS 的值，从而使卡的电流小于等于 200mA。 1：SDIO 卡总电流可以超过 200mA，且 SPS 和 EPS 有效。主机根据自己提供电流的能力使用 FBR 中的 SPS、EPS 以及 IOEx 使能更大电流的 function。	1'b0
			RO	[7:2],RFU	
0X13	High- Speed	SHS	RO	[0]，表示 SDIO 卡支持高速 0：不支持高速 1：支持高速	1'b1

		EHS	RW	[1]，高速使能 0：SDIO 卡工作在缺省速度下，最高频率 25MHZ 1：SDIO 卡可以工作在高速模式下，最高频率 50MHZ	1’b0
			RO	[7-2]，RFU	
0X14- 0XEF	RFU		RO	Reserved for Future Use (RFU)	8’b0
0XF0- 0XFF	Reserved for Vendors		RO	Area Reserved for Vendor Unique Registers	8’b0
0X100	I/O Device Interface Code		RO	[3:0],标志 Fn1 为什么样的设备。 通过寄存器 CIA[15:12]可编程。 4’h0 No SDIO standard interface supported by this function 4’h1 This function supports the SDIO Standard UART 4’h2 This function supports the SDIO Type-A for Bluetooth standard interface 4’h3 This function supports the SDIO Type-B for Bluetooth standard interface 4’h4 This function supports the SDIO GPS standard interface 4’h5 This function supports the SDIO Camera standard interface 4’h6 This function supports the SDIO PHS standard interface	4’h7

			4’h7 This function supports the SDIO WLAN interface 4’h8 This function supports the Embedded SDIO-ATA standard interface	
	RFU	RO	[5:4],RFU	2’b00
	Function supports CSA	RO	[6], 0：Fn1 不支持 CSA 1：Fn1 支持且有 CSA 可以通过寄存器 CIA[16]编程	1’b0
	Function CSA enable	RW	[7], 0:不允许访问 CSA 1：允许访问 CSA	1’b0
0X101	Extended standard I/O device type code	RO	[7:0]，I/O Device Interface Code 的扩展 通过可以通过寄存器 CIA[24：17]编程	8’b0
0X102	SPS	RO	[0],标明 Fn1 是否有功耗选择 0：没有功耗选择 1：有两个功耗选择，可以通过 EPS 选择 通过可以通过寄存器 CIA[25]编程	1’b0
	EPS	RW	[1],功耗选择 0：Fn1 工作在高电流模式 1：Fn1 工作在低电流模式	1’b0

		RO	[7：2]，RFU	6’b0
0X103- 0X108		RO	RFU	0
0X109- 0X10B	Address pointer to function CIS1	RO	[16:0],Fn1 的 CIS 地址指针，即 CIS1，指示主机访问 Fn1 的 CIS 起始地址。存储方式为 LSB 小段格式。	17’h02000
		RO	[23：17],RFU	7’b0
0X10C - 0X10E	Address pointer to function CSA	RW	[23:0],指向 CSA 的 24bit 地址指针，主机通过 CSA 访问窗口访问 CSA 后，该指针自动加 1. 地址采用小段格式存储（LSB）	24’h000000
0X10F	Data access window to CSA	RW	[7:0]，对 CSA 的读写窗口。对该地址写操作时，对应的数据会通过本窗口写入 CSA 24bit 地址指针所指示的地址中，对该地址读操作时，从 24bit CSA 地址指针所指示的地址读取数据，通过本窗口送给主机。	8’b00
0X110- 0X111	Function1 IO Block Size	RW	[15:0]，16bit 的寄存器，用来设置 IO block size。最大的 block size 为 2048Byte，最小为 1。 该数据以小端模式存储（LSB）。	16’b0
0X1000- 0X1010	CIS0	RO	主机访问 Fn0 的 CIS 地址空间，即主机通过这个地址空间段访问 CIS0.本 SDIO 卡中支持最多 17 个字节的 CIS0	
0X2000-	CIS1	RO	主机访问 Fn1 的 CIS 地址空间，即主机通过这个地址空间段访问 CIS1。本 SDIO 卡支持 CIS1 的字节数据为 55~308.	

0X2133				
RFU			RFU	

11.4.3 SDIO Fn1 寄存器

Fn1 寄存器为 SDIO 协议分配给 function1 的地址空间，其地址范围为：0x00000~0x1FFFF，共 128K。由于芯片内部 AHB 总线地址位宽为 32 位，SDIO 无法使用 17 位地址直接对芯片内部进行访问，因此在设计中，需要完成一次地址映射。具体映射关系如下表：（FN1 访问空间）

表 91 SDIO Fn1 地址映射关系

SDIO 主机访问窗口	对应实际物理地址空间	实际物理地址空间内容
0X0000 ~ 0X00FF	0X0000 ~ 0X00FF	SDIO 模块内部寄存器地址空间。
0X1000 ~ 0X1FFF	可配置	CIS0 物理空间，具体物理空间由固件配置。
0X2000 ~ 0X2FFF	可配置	CIS1 物理空间，具体物理空间由固件配置。
0X4000 ~ 0X4FFF	可配置	下行和上行 cmd 物理空间，具体物理空间地址由固件配置。
0X5000 ~ 0X5FFF 0X15000 ~ 0X15FFF	可变	发送 buffer 地址空间，具体根据 sdio_txbd 中的指示。
0X6000 ~ 0X7FFF 0X16000 ~ 0X17FFF	可变	接收 buffer 地址空间，具体根据 sdio_rxbd 中的指示。
0X8000 ~ 0X9FFF	0X0E000000 ~ 0X0E002000	AHB 总线 config 地址空间。
0XA000 ~ 0XBFFF	0X0F000000 ~ 0X0F002000	AHB 总线 APB 地址空间。

驱动应该避免访问超出以上范围的空间，这样做可能会带来意想不到的结果。

其中第一项地址空间寄存器是在 SDIO 内部，并且只能由 SDIO HOST 访问；其他地址空间的访问将根据描述映射到芯片内部其他空间中。

本节只介绍 SDIO 0x0000 ~ 0x00FF 地址空间中的寄存器，这些寄存器由 SDIO 主机通过 CMD52 命令直接访问，偏移地址即为访问地址，功能号为 1。

表 92 SDIO Fn1 部分寄存器（供 HOST 访问）

偏移地址	名称	位宽	访问	描述	复位值
0X00~0X03			RO	RSV	
0X04		[7:1]	RO	RSV	7'b0
	int_read_data	[0]	RW	上行数据中断。高有效，写 1 清零。 在读取 0x1C 时，也会被自动清 0。	1'b0
0X05		[7:1]	RO	RSV	7'b0
	int_mask	[0]	RW	对应 int_src 的屏蔽使能信号。1 则屏蔽相应中断。	1'd0
0X06		[7:2]	RO	RSV	6'b0
	wlan_awake_stts	[0]	RO	当前 WLAN 状态： 1 为 ACTIVE；0 为 SLEEP。	1'b1
0X1C	dat_len0	[6:0]	RO	上行数据长度高 7bit。共 12bit，低 5bit 在 0x1D 中。	7'b0
	dat_vld	[7]	RO	1'b1	1'b1

0X1D	dat_len1	[7:3]	RO	上行数据长度低 5bit。共 12bit，高 7bit 在 0x1C 中。	5'b0
		[2:0]	RO	RSV	3'b0
0X1E			RO	RSV	
0X1F		[7:2]	RO	RSV	6'b0
	down_cmdbuf_vld	[1]	RO	下行命令 buffer 可用，1 有效。	1'b1
	txbuf_vld	[0]	RO	下行数据 buffer 可用。1 有效，表示存在可用的发送 buffer。	1'b0
0X20	wlan_wake_en	[0]	RW	SLEEP 状态下 SDIO 下发的芯片唤醒使能，高有效。芯片被唤醒后，此位将被硬件自动清 0。	1'b0
0X21		[7:1]	RO	RSV	7'b0
	fn1_rst	[0]	RW	软复位，1 有效。 软件写 1 后，（即芯片 wlan 部分电路）被复位，写 0 后，function1 复位被释放。	1'b0
0X22		[7:1]	RO	RSV	7'b0
	fn1_recov	[0]	RW	错误恢复使能，1 有效，命令 response 结束后，该位自动清 0。 该功能用来完成 fn0/fn1 io abort 相同的功能，当存在 cmd 异常或者提前结束命令时，可以将此寄存器置 1，来完成 io abort 操作。 由于在某些 bus driver 版本中，io abort 命令受限制	1'b0

				（即该寄存器访问地址空间受限），不允许用户驱动使用，此时向该寄存器写 1 可以取代 io abort 操作，并产生相同的效果。	

11.4.3.1 SDIO AHB 接口从设备寄存器

下面的寄存器，在 SDIO 从设备初始化的时候使用。

传输数据时，需要和 wrapper 控制器配合使用，Wrapper 控制器的部分参考 HSPI 部分说明文档。

表 93 SDIO AHB 总线寄存器

偏移地址	名称	位宽	访问	描述	复位值
0X0000 0X0004			RO	Rsv	
0X0008	CIS function0 address	[31:0]	RW	CIS0 在系统内部 memory 中存储的偏移地址。 CIS0 实际存储起始地址 =0x01000（读命令起始地址）+该偏移地址	32’b0
0X000C	CIS function1 address	[31:0]	RW	CIS1 在系统内部 memory 中存储的偏移地址。 CIS1 实际存储起始地址 =0x02000（读命令起始地址）+该偏移地址	32’b0
0X0010	CSA address	[31:0]	RW	固件初始化时设置访问 CSA 的便宜地址，其原理和 CIS 设置相同。本设计中不支持 CSA。	32’b0
0X0014	Read address	[31:0]	RW	用来设置 DMA 从 memory 读取数据的起始地址。配	32’b0

				合 Data Port 寄存器可以实现对系统内部 memory 的读操作（即访问地址为 0x00+ Read address）。在本设计中，没有采用这种方法，因此缺省为 0	
0X0018	Write address	[31:0]	RW	用来设置 DMA 向 memory 写数据的起始地址。配合 Data Port 寄存器可以实现对系统内部 memory 的写操作（即访问地址为 0x00+ write address）。在本设计中，没有采用这种方法，因此缺省为 0	32'b0
0X001C	AHB Transfer count	[20:0]	RW	SDIO 设备通知主机，在将发起的读数据操作中，需要读取多少字节数据。 [23:21]RFU	32'b0
0X001F		RO	--	rsv	
0X0020	SDIO Transfer count	[20:0]	RO	一次数据传输中，从主机向 SDIO 设备下发的字节数。当数据传输完成后，通过该寄存器高速设备内部下发的字节数。 [31:21]RFU	32'b0
0X0024	CIA register	--	RW	[3:0]:CCCR Revision。缺省为 4'h2 4'h0 CCCR/FBR Version 1.00 4'h1 CCCR/FBR Version 1.10 4'h2 CCCR/FBR Version 1.20 [7:4]SDIO Revision。缺省为 4'h3 4'h0 SDIO Specification 1.00	32'h06017232

				4'h1 SDIO Version 1.10 4'h2 SDIO Version 1.20 4'h3 SDIO Version 2.0 [11:8]SD Revision。缺省为 4'h2 4'h0 SD Physical Specification 1.01 4'h1 SD Physical -Spec-1.10 4'h2 SD Physical Spec 2.0 [15:12]IO-Device Code。缺省为 4'h7，即本产品为 WLAN 设备。 [16]csa_support,缺省为 1. 0：不支持 CSA 1：支持 CSA 初始化时，固件需要配置该寄存器为 0。 [24:17],Extended IO-device code 缺省为 8'b0。 Fn1 的 standard IO-device code 的扩展。 [25],SPS,缺省为 1，即 Fn1 支持高功耗 0：不支持	

				1：支持 [26],SHS,缺省为 1，支持高速 0：不支持 1：支持 [31:27]:RFU	
0X0028	Program Register	--	RW	[0],function ready。hc8051 在完成 SD 初始化所需要的 Fn1 寄存器配值后，置位该寄存器，向 SD 主机标明 Fn1 已经准备好工作。缺省为 0 0:Fn1 not ready 1：Fn1 ready [1]，fun1 read data ready。Fn1 准备好向 SD 主机送数据时，置位该寄存器。当主机响应读中断后，读中断源寄存器（Interrupt Identification），该 bit 自动清零。缺省为 0. 0：无数据送给主机 1：有数据送给主机。 [2]，SCSI。支持连续的 SPI 中断。缺省为 1. 0：不支持 1：支持 [3],SDC。标明 SDIO 卡在数据传输同时，支持执行 CMD52 命令。缺省为 1 0：不支持	16'h 02fc

				1：支持 [4],SMB。SDIO 卡支持 CMD53 的 Block 传输。缺省为 1. 0：不支持 1：支持 [5],SRW。标明 SDIO 卡支持读等待。缺省为 1. 0：不支持 1：支持 [6],SBS。标明 SDIO 卡支持挂起/恢复。缺省为 1 0：不支持 1：支持 [7],S4MI。标明 SDIO 卡支持在 4bit 多 Block 数据传输时产生中断。缺省为 1。 0：不支持 1：支持 [8],LSC。标明 SDIO 卡是低速设备。缺省为 0. 0：全速设备 1：低速设备 [9]，4BLS。标明 SDIO 卡是一个低速设备，但支持 4bit 数据传输。缺省为 1。 0：不支持 1：支持	

				[10],card ready。属于 SD 时钟域的信号，上电复位释放后，该寄存器自动变为 1，标明 SDIO（接口部分）已经准备好。固件监测到该信号，即可以配置初始化时需要的 Fn1 寄存器。上电复位时为 0. [15：11],RFU。缺省为 0	
0X0034	OCR register	--	RW	[23:0]，工作情况寄存器，内部可编程，主要用来与主机工作电压范围匹配。缺省为：24'hff8000。 寄存器 Bit 支持电压范围 0-3 Reserved 4 Reserved 5 Reserved 6 Reserved 7 Reserved 8 2.0-2.1 9 2.1-2.2 10 2.2-2.3 11 2.3-2.4 12 2.4-2.5 13 2.5-2.6 14 2.6-2.7 15 2.7-2.8	32'h00ff8000

				16 2.8-2.9 17 2.9-3.0 18 3.0-3.1 19 3.1-3.2 20 3.2-3.3 21 3.3-3.4 22 3.4-3.5 23 3.5-3.6 [31:24]，8’b0	
0X0038		RW	--	rsv	32’h0
0X003C	CD_State Register	--	RW	[0]，标明 SD dat[3]数据线上拉电阻的状态。 0：上拉有效 1：上拉无效 片内 AHB 总线都可以对该寄存器访问。 注：AHB 对该寄存器的写操作结果会间接修改 CCCR7[7] CD 的值。 [31:1],RFU	32’b0
0X0040	Fn1_Ena Register	--	RW	[0],标明 Fn1 是否被使能。 0：禁止 1：使能	32’b0

				片内 AHB 总线都可以对该寄存器访问。 注：AHB 对该寄存器的写操作结果会间接修改 CCCR1[1] IOE1 的值。 [31:1],RFU	

12 HSPI/SDIO Wrapper 控制器

12.1功能概述

配合接口控制器（SDIO 和 HSPI）完成主机和芯片内部缓存之间数据的 DMA 操作。包括上下行数据缓存的软硬件交互控制，发送与接收缓存的填充与释放，上行数据中断的产生等等。

SDIO 和 HSPI 都通过 wrapper 控制器和上位机交互数据，两者的控制命令以及流程是相同的。为了描述方便，部分寄存器加上了前缀 SDIO。相应寄存器操作同样适用于 HSPI

需要注意的是，对于前缀是 SDIO_TX 相关寄存器，控制的是设备接收数据，对于 SDIO_RX 相关寄存器，控制的是设备发送数据。

对于寄存器中出现的 cmd 字段，只是从描述上和数据帧区分，并不表示命令通道只能传输命令，也可以用来传输数据。在这里命令和数据的区别是，命令通道只有一块缓存区，并且缓存长度一般会少于 256 字节，而数据通道有多块缓存区，每块缓存区的长度都超过 1K，多块缓存构成一个链表结构，具体长度由软件配置。由于数据帧的这种链表缓存结构，传输速率会快于命令通道。

12.2主要特性

- 支持字对齐的数据搬移
- 支持 DMA 功能
- 支持链表结构管理
- 支持中断产生
- 最大可接收 4096 字节数据

12.3 功能描述

12.3.1 上行数据接收功能

上行方向是指主设备（SDIO 或者 HSPI）向从设备（W800）发送数据的方向。

当主设备向从设备发送数据时，SDIO 或者 HSPI 模块接收到数据后，会通过 WRAPPER 把数据链接到接收 BD，并产生中断通知应用软件处理数据。

接收 BD 描述符：

sdio_rxbd

31		0	
Vld[31]	RSV		word0
Sdio_rx_info_size[31:26]	Frm_len[25:12]	Rxbuf0_offset[11:0]	word1
Sdio_rxbuf0_addr[31:0]			word2
Sdio_rxbuf1_addr[31:0]			word3
Sdio_rxbuf2_addr[31:0]			word4
Next_sdio_rxbd_addr[31:0]			word5

说明：
1. vld，1为有效，表示当前描述符指向有效的接收帧。
2. rxbuf0_offset：字节地址，字对齐，表示802.3帧第一个有效字节所在的字相对于sdio_rxbuf0的偏移地址。
3. sdio_rxbuf0_addr：字节地址，字对齐，上行数据帧的第一个分片的buf基地址。
4. sdio_rxbuf1_addr：字节地址，字对齐，上行数据帧的第二个分片的buf基地址。
5. sdio_rxbuf2_addr：字节地址，字对齐，上行数据帧的第三个分片的buf基地址。
6. next_sdio_rxbd_addr，字节地址，字对齐，下一个sdio_rxbd的基地址。
7. frm_len，字节长度，表示上行数据长度，不包括sdio_rx_info。
8. Sdio_rx_info_size，字节长度，表示sdio_rx_info字节个数，必须是4字节整数倍。

注意：接收最长帧为4096，最多占用三个分片。Rxbuf0_offset只对存放帧的第一个分片有效（即sdio_rxbuf0中的分片），对剩余的两个分片，数据从基地址顺序存放。硬件应根据上行数据长度和每个buf固定1600字节大小来确定此帧是否存在于多个分片之中。

图 26 SDIO 接收 BD 描述符

当 W800 的 SDIO 模块或者 HSPI 模块检测到接收使能有效后，读取 RXBD，并判断 Vld 的标志。

12.3.2 下行数据搬移功能

当 W800 有数据要向主设备发送数据的时候，软件先把发送描述准备好，然后，通知 WRAPPER 把数据搬移,WRAPPER 通过 SDIO 或者 HSPI 的中断信号通知主设备来读取要发送的设备,当数据发送完成后，WRAPPER 产生发送完成中断通知程序。

发送 BD 描述符：

说明：
1.vld，1为有效，表示当前描述符指向可用的发送缓存。
2.Sdio_txbuf_addr，字节地址，必须字对齐。指示本描述符指向发送数据存放的基地址。该地址相对于每个sdio_txbuf的基地址有一定的偏移，以保证在固件在帧的开始处向上添加LLC时不会超越sdio_txbuf的基地址。
3.next_sdio_txbd_addr，字节地址，字对齐。下一个sdio_txbd的基地址。

图 27 SDIO 发送 BD 描述符

12.4寄存器描述

12.4.1 寄存器列表

表 94 WRAPPER 控制器寄存器

偏移地址	名称	缩写	访问	描述	复位值
0X0000	WRAPPER 中断状态寄存器	INT_STTS	RW	命令或者数据帧中断状态	0X0000

0X0004	WRAPPER 中断配置寄存器	INT_MASK	RW	命令或者数据帧中断是否屏蔽	0X0000
0X0008	WRAPPER 上行命令就绪寄存器	UP_CMD_AVAIL	RW	上行命令是否就绪	0X0000
0X000c	WRAPPER 下行命令 buf 就绪寄存器	DOWN_CMD_BUF_AVAIL	RW	下行命令 buf 是否就绪	0X0000
0X0010	SDIO_TX 链接使能寄存器	SDIO_TX_BD_LINK_EN	RW	指示 sdio_txbd 链表描述符是否有效	0X0001
0X0014	SDIO_TX 链接地址寄存器	SDIO_TX_BD_ADDR	RW	当前 sdio_txbd 指向的地址，需要字对齐，初始化时需要配置	0X0000
0X0018	SDIO_TX 使能寄存器	SDIO_TX_EN	RW	SDIO 发送帧使能	0X0000
0X001c	SDIO_TX 状态寄存器	SDIO_TX_STTS	RO	SDIO 发送状态	0X0000
0X0020	SDIO_RX 链接使能寄存器	SDIO_RX_BD_LINK_EN	RW	指示 sdio_rxbd 链表描述符是否有效	0X0001
0X0024	SDIO_RX 链接地址寄存器	SDIO_RX_BD_ADDR	RW	当前 sdio_rxbd 指向的地址，需要字对齐，初始化时需要配置	0X0000
0X0028	SDIO_RX 使能寄存器	SDIO_RX_EN	RW	SDIO 接收帧使能	0X0000
0X002c	SDIO_RX 状态寄存器	SDIO_RX_STTS	RO	SDIO 接收状态	0X0000
0X0030	WRAPPER CMD BUF 基地址寄存器	CMD_BUF_BASE_ADDR	RW	下行 cmd buf 基地址	0X0000
0X0034	WRAPPER CMD BUF SIZE 寄存器	CMD_BUF_SIZE	RW	Cmd buf 字节大小	0X0064

12.4.2 WRAPPER 中断状态寄存器

表 95 WRAPPER 中断状态寄存器

位	访问	操作说明	复位值
[31: 4]	RO	保留	
[3]	RW	int_down_cmd 下行 cmd 帧完成中断。写 1 清 0。	1'b0
[2]	RW	int_up_cmd 上行 cmd 帧字完成中断。写 1 清 0。	1'b0
[1]	RW	int_sdio_txfrm 下行数据帧完成中断。写 1 清 0。	1'b0
[0]	RW	int_sdio_rxfrm 上行数据帧完成中断。写 1 清 0。	1'b0

12.4.3 WRAPPER 中断配置寄存器

表 96 WRAPPER 中断配置寄存器

位	访问	操作说明	复位值
[31: 4]	RO	保留	
[3]	RW	int_mask_down_cmd 下行 cmd 帧完成中断屏蔽寄存器。1 为屏蔽，下同。	1'b0
[2]	RW	int_mask_up_cmd 上行 cmd 帧完成中断屏蔽寄存器。	1'b0
[1]	RW	int_mask_sdio_txfrm 下行数据帧完成中断屏蔽寄存器。	1'b0
[0]	RW	int_mask_sdio_rxfrm 上行数据帧完成中断屏蔽寄存器。	1'b0

12.4.4 WRAPPER 上行命令就绪寄存器

表 97 WRAPPER 上行命令就绪寄存器

位		访问	操作说明	复位值

[31: 1]		RO	保留	
[0]		RW	存在上行 cmd 帧时，固件置此位为 1。当完成上行 cmd 传输时，硬件自动将其清 0，并产生 int_up_cmd 中断。	1'b0

12.4.5 WRAPPER 下行命令 buf 就绪寄存器

表 98 WRAPPER 下行命令 buf 就绪寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	发送完下行 cmd 后，硬件将此位清 0，同时产生中断。当固件处理完成此下行命令后，将此位置 1。	1'b0

12.4.6 SDIO TX 链接使能寄存器

表 99 SDIO TX 链接使能寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	sdio_txbd 链接使能，高有效。 若此位有效，硬件在处理完成一个 sdio_txbd 描述符，直接处理 next_sdio_txbd_addr 指向的下一个描述符。若此位无效，硬件在处理完成一个 sdio_txbd 后，会立即停止。 对 HSPI 同样适用。	1'b1

12.4.7 SDIO TX 链接地址寄存器

表 100 SDIO TX 链接地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	当前 sdio_txbd 字节地址，软件需要严格保证字对齐，下同。 初始时固件需要配置此寄存器，硬件每次处理完成一个发送 buf 后，将 sdio_txbd 描述符中的 next_sdio_txbd_addr 更新至此寄存器。 对 HSPI 同样适用。	32'h0

12.4.8 SDIO TX 使能寄存器

表 101 SDIO TX 使能寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	SDIO 发送帧使能，高有效。 固件在组建完成每个描述符 sdio_txbd 描述符后，将此位置 1，以通知 SDIO 模块存在新的发送描述符。SDIO 模块检测到此位有效，则启动读取当前的 sdio_txbd，并完成发送帧流程。 硬件自动完成此寄存器的清 0 操作。 对 HSPI 同样适用。	1'b0

12.4.9 SDIO TX 状态寄存器

表 102 SDIO TX 状态寄存器

位	访问	操作说明	复位值

[31: 1]	RO	保留	
[0]	RW	SDIO 发送状态。 0：SDIO 由于没有可用发送描述符，已停止发送流程 1：SDIO 处于发送过程中 对 HSPI 同样适用。	1’b0

12.4.10SDIO RX 链接使能寄存器

表 103 SDIO RX 链接使能寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	sdio_rxbd 链接使能，高有效。 若此位有效，硬件在处理完成一个 sdio_rxbd 描述符，直接处理 next_sdio_rxbd_addr 指向的下一个描述符。若此位无效，硬件在处理完成一个 sdio_rxbd 后，会立即停止。 对 HSPI 同样适用。	1’b1

12.4.11 SDIO RX 链接地址寄存器

表 104 SDIO RX 链接地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	当前 sdio_rxbd 字节地址。 初始时固件需要配置此寄存器，硬件每次处理完成一个发送 buf 后，将 sdio_rxbd 描述符中的 next_sdio_rxbd_addr 更新至此寄存器。	32’h0

		对 HSPI 同样适用。	

12.4.12SDIO RX 使能寄存器

表 105 SDIO RX 使能寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	SDIO 接收帧使能，高有效。 固件在组建完成每个描述符 sdio_rxbd 描述符后，将此位置 1，以通知 SDIO 模块存在新的接收描述符。SDIO 模块检测到此位有效，则启动读取当前的 sdio_txbd，并完成接收帧流程。 硬件自动完成此寄存器的清 0 操作。 对 HSPI 同样适用。	1'b0

12.4.13 SDIO RX 状态寄存器

表 106 SDIO RX 状态寄存器

位	访问	操作说明	复位值
[31: 1]	RO	保留	
[0]	RW	SDIO 接收状态。 0：SDIO 由于没有有效的上行描述符和上行命令，已停止接收流程 1：SDIO 处于接收过程中 对 HSPI 同样适用。	1'b0

12.4.14 WRAPPER CMD BUF 基地址寄存器

表 107 WRAPPER CMD BUF 基地址寄存器

位	访问	操作说明	复位值
[31: 0]	RW	下行 cmd buf 基地址。 上行 cmd buf 基地址为该基地址加上 cmd_buf_size。	32'h0

12.4.15 WRAPPER CMD BUF SIZE 寄存器

表 108 WRAPPER CMD BUF SIZE 寄存器

位	访问	操作说明	复位值
[31:12]	RO	保留	
[11: 0]	RW	cmd buf 字节大小，必须为 4 字节的整数倍。	12'd64

13 SDIO HOST 设备控制器

13.1 功能概述

SDIO HOST 设备控制器提供了一个能够访问安全数字输入输出卡（SDIO）以及 MMC 卡的数字接口。能够访问兼容 SDIO 2.0 协议的 SDIO 设备和 SD 卡设备。主要接口有 CK，CMD 以及 4 根数据线。

13.2 主要特性

- 兼容 SD 卡规范 1.0/1.1/2.0(SDHC)
- 兼容 SDIO 内存卡规范 1.1.0
- 兼容 MMC 规范 2.0~4.2
- 可配置接口时钟速率，支持主机速率 0~50MHz，
- 支持标准 MMC 接口
- 支持最大 1024 字节的 Block
- 支持软复位功能
- 自动 Command/Response CRC 生成/校验；
- 自动数据 CRC 生成/校验；
- 可配置 timeout 检测；
- 支持 SPI、1 比特 SD 和 4 比特 SD 模式
- 支持 DMA 数据传输

13.3 功能描述

13.4寄存器描述

13.4.1 寄存器列表

偏移地址	寄存器名称	名称	位宽	属性	说明	复位值
0x00	mmc_ctrl	RSV	[15:11]	RO		
			[10]	RW	SDIO 读等待使能 ‘1’：使能 SDIO 读等待 ‘0’：关闭 SDIO 读等待	1’b0
			[9]	RW	SDIO 中断使能 ‘1’：SDIO 中断使能 ‘0’：SDIO 中断关闭	1’b0
			[8]	RW	SDIO 模式使能 ‘1’：SDIO ‘0’：SD/MMC	1’b0
			[7]	RW	SD/MMC/SDIO 接口数据宽度 ‘1’：4 bits ‘0’：1 bit	1’b0
			[6]	RW	SD/MMC/SDIO 传输模式 ‘1’：High-Speed Mode ‘0’：Low-Speed Mode	1’b1
			[5:3]	RW	SDIO/MMC/SDIO 端口时钟速率选择 参考 Table 2	3’b000

			[2]	RW	SDIO/MMC/SDIO 接口驱动模式选择 '1' : open_DrainMode '0' :Push-Pull Mode	1'b1
			[1]	RW	信号模式选择 '1' : 自动选择传输模式 '0' :使用 mmc_port 寄存器选择	1'b0
			[0]	RW	端口模式选择 '1' : MMC mode '0' :SPI mode	1'b1
0x04	mmc_io	RSV	RO	[15:10]		6'd0
			RW	[9]	SDIO cmd12/IO Abort 标志 '1' : 将当前命令标志为 cmd12/IO Abort '0' : 标志当前命令不是 cmd12/IO Abort	1'b0
			RW	[8]	SDIO 命令属性 '1' : 标记当前命令后有数据块; '0' : 标记当前命令后没有数据块和命令响应;	1'b0
			RW	[7]	Enable auto generate 8 null clock after response/command or single block data	1'b0

					在响应/命令或者单个数据块后自动生成 8 个 null 时钟功能 '1'：使能 '0'：关闭	
			RW	[6]	Enable auto receive response after command 在命令后自动接收响应功能 '1'：使能 '0'：关闭	1'b0
			RW	[5]	SD/MMC/SDIO 端口时钟线上 8 个 null 时钟生成 '1'： 生成 8 个 null 时钟 '0'：根据 bit 3 设置 接收响应/发送命令	1'b0
			RW	[4]	为 CID 和 CSD 读设计。当发送 CID 或者 CSD 命令时，SD/MMC/SDIO 卡设备会在 CMD 线上回复 136bit CID 或者 11 CSD 数据。当设置此 bit 为 1 时，CID 或者 CSD 数据将存储在命令 buffer 区域的[135:8]中；	1'b0
			RW	[3]	bit[5] 为'0'时响应/命令选择 '1'：接收响应	1'b0

					‘0’：发送命令.	
			RW	[2]	设置 自动 8 null 时钟/命令/响应传输功能 ‘1’：使能自动 8 null 时钟/命令/响应传输 ‘0’：关闭自动 8 null 时钟/命令/响应传输. 根据 bit 5 与 bit3 设置，生成 8 个 null 时钟，接收 response 或者传输命令。当传输完成，此 bit 自动清除；	1’b0
			RW	[1]	设置数据传输方向 ‘1’：读数据；. ‘0’：写数据；	1’b0
			RW	[0]	设置自动数据传输 ‘1’：使能自动数据传输 ‘0’:关闭自动数据传输. 当数据传输完毕，此 bit 将被自动清除；	1’b0
0x08	mmc_bytecntl		RW	[15:0]	数据传输字节计数寄存器	16’h0200
0x0C	mmc_tr_blockcnt		RO	[15:0]	多数据块传输时，已完成数据块计数器	16’h0000
0x10	mmc_crcctl		RW	[7]	SD/MMC/SDIO port CMD Line CRC circuit enable. SD/MMC/SDIO 端口 CMD 线 CRC 功	1’b0

					能 ‘1’：使能. ‘0’：关闭.	
			RW	[6]	SD/MMC/SDIO 端口 数据 线 CRC 功能 ‘1’： 使能. ‘0’： 关闭.	1’b0
			RW	[5]	使能自动 CRC 检查 crc_status ‘1’：使能，当 crc_status !=3’b010 时，会产生 crc 状态中断，写数据传输将被 stop 命令中断，并且 mmc_io[0] 或者 mmc_io_mbctl[2:0] 将被清除； 0：关闭；	1’b0
			RW	[4]	在 response 之前读多数据块功能 ‘1’：使能. ‘0’：关闭	1’b0
			RW	[3:2]	DAT CRC selection. DAT CRC 选择 参考 Table 4	1’b0
			RO	[1]	CMD CRC 错误.	1’b0
			RO	[0]	DAT CRC 错误	1’b0
0x14	cmd_crc	RSV	RO	[7]	RSV	1’b0
			RO	[6:0]	CMD CRC 寄存器值	7’d0

0x18	dat_crcl		RO	[7:0]	The DAT CRC 低位寄存器值	NA
0x1C	dat_crch		RO	[7:0]	The DAT CRC 高位寄存器值	NA
0x20	mm_port		RW	[7]	SD/MMC/SDIO port 时钟线信号.	1'b0
			RW	[6]	SD/MMC/SDIO port CMD 线信号	1'b1
			RW	[5]	SD/MMC/SDIO port 数据线信号	1'b1
			RW	[4]	自动 检查 Ncr 超时功能 1': 使能自动检查 Ncr 超时. '0' :关闭自动检查 Ncr 超时	1'b1
			RW	[3:0]	Ncr 超时计数值 (SD/MMC/SDIO 时钟数).	4'hF
0x24	mmc_int_mask	RSV	RO	[15:9]		7'd0
				[8]	SDIO 数据线 1 中断 屏蔽 '1': 不屏蔽 '0': 屏蔽	1'b0
				[7]	CRC 状态 token 错误中断屏蔽 '1': 不屏蔽 '0': 屏蔽	1'b0
				[6]	命令和响应 Ncr 超时中断屏蔽 '1': 不屏蔽 '0': 屏蔽	1'b0
				[5]	多数据块超时中断 屏蔽. '1': 不屏蔽	1'b0

					‘0’：屏蔽	
				[4]	多数据块传输完成中断 屏蔽. ‘1’：不屏蔽 ‘0’：屏蔽	1’b0
				[3]	命令 CRC 错误中断屏蔽 ‘1’：不屏蔽 ‘0’：屏蔽	1’b0
				[2]	数据 CRC 错误中断屏蔽 ‘1’：不屏蔽 ‘0’：屏蔽	1’b0
				[1]	数据传输完成中断屏蔽 ‘1’：不屏蔽 ‘0’：屏蔽	1’b0
				[0]	命令传输完成中断屏蔽 ‘1’：不屏蔽 ‘0’：屏蔽	1’b0
0x28	clr_mmc_int	RSV	RO	[15:9]		7’d0
			RW	[8]	W: 清除 SDIO 数据线 1 中断 R: SDIO 数据线 1 中断状态	1’b0
			RW	[7]	W: 清除 CRC 状态 token 错误 中断 R: CRC 状态 token 错误 中断状态;	1’b0

					当此 bit 为 1 时，判断 mmc_sig[6:4]	
			RW	[6]	W：清除命令和响应 Ncr 超时中断； R：命令和响应 Ncr 超时中断状态；	1'b0
			RW	[5]	W：清除多数据块超时中断；. R：多数据块超时中断状态；	1'b0
			RW	[4]	W：清除多数据块传输完成中断；. R：多数据块传输完成中断状态；	1'b0
			RW	[3]	W：清除命令 CRC 错误中断；. R：命令 CRC 错误中断状态；	1'b0
			RW	[2]	W：清除数据 CRC 错误中断；. R：数据 CRC 错误中断状态；	1'b0
			RW	[1]	W：清除数据传输完成中断；. R：数据传输完成中断状态；	1'b0
			RW	[0]	W：清除命令传输完成中断；. R：命令传输完成中断状态；	1'b0
0x2C	mmc_cardsel		RW	[7]	SD/MMC/SDIO 控制器使能 '1'：使能	1'b0

					‘0’：关闭	
			RW	[6]	使能 SD/MMC/SDIO 卡设备时钟线 ‘1’：使能 ‘0’：关闭	1’b1
			RW	[5:0]	SD/MMC/SDIO 时间基准系数 使用此寄存器建立 1MHz 时钟； 1MHz=Fhclk/((mmc_cardsel[5:0]+1)*2)	6’d0
0x30	mmc_sig		RW	[7]	SD/MMC/SDIO port CMD 线信号 当 读取此寄存器时，SDIO 控制器将在时钟线上生成一个时钟脉冲。并且在时钟上升沿时 CMD 线上状态将被存储到此寄存器中。	1’b1
			RW	[6:4]	CRC status[2:0] 当写数据 CRC status token 时；	3’b111
			RW	[3]	SD/MMC/SDIO port DAT3 数据信号.	1’b1
			RW	[2]	SD/MMC/SDIO port DAT2 数据信号.	1’b1
			RW	[1]	SD/MMC/SDIO port DAT1 数据信号.	1’b1
			RW	[0]	SD/MMC/SDIO port DAT0 数据信号.	1’b1
0x34	mmc_io_mbctl		RW	[7:6]	SD/MMC/SDIO NAC 超时范围选择	2’b0

					参考 Appendix 2 -Table 5.	
			RW	[5:4]	SD/MMC/SDIO Busy timeout scale selection. SD/MMC/SDIO 设备忙超时范围选择. 参考 Appendix 2 -Table 6	2'd1
			RW	[3]	SD/MMC/SDIO port 时钟线极性 1：时钟下降沿发送，上升沿采集； 0：时钟上升沿发送，下降沿采集；	1'b0
			RW	[2]	设置 SD/MMC/SDIO 端口全自动命令和多数据块传输 '1'：使能 '0'：关闭 设置此 bit 为 1（mmc_io[7:6]==11）会触发一个 SD/MMC/SDIO 的命令，响应，8 null 时钟，多数据块传输。当数据传输完毕，此 bit 会自动清零。	1'b0
			RW	[1]	Select multiple block data transfer direction. 设置多数据块传输方向 '1'：读数据. '0'：写数据.	1'b0

			RW	[0]	Set SD/MMC/SDIO port auto multiple block data ransfer. 设置 SD/MMC/SDIO 端口自动多数据块传输 '1'：使能. '0'：关闭. 设置此 bit 为 1（mmc_io[7:6]==11）会触发一个 SD/MMC/SDIO 的多数据块传输。数据块个数在 mmc_blocknt 寄存器中设置。当数据传输完毕，此 bit 会自动清零。	1'b0
0x38	mmc_blockcnt		RW	[15:0]	Data block number register. 数据块个数寄存器 配置此寄存器定义在多数据块传输中总共传输数据块个数。	16'h0001
0x3C	mmc_timeoutcnt		RW	[7:0]	Data transfer timeout count register. 数据传输超时计数器 Time = Scale* bit[7:0]. Scale 通过寄存器 mmc_io_mbctl[7:6]/[5:4] 定义；	8'h40
0x40	cmd_buf0		RW	[7:0]	The cmd_buf byte 0. Mapped to command line bit [15:8]	8'h00

					命令 buf 字节 0，映射到命令线上 bit[15:8]	
0x44	cmd_buf1		RW	[7:0]	The cmd_buf byte 1. Mapped to command line bit [23:16] 命令 buf 字节 1，映射到命令线上 bit[23:16]	8'h00
0x48	cmd_buf2		RW	[7:0]	The cmd_buf byte 2. Mapped to command line bit [31:24] 命令 buf 字节 2，映射到命令线上 bit[31:24]	8'h00
0x4C	cmd_buf3		RW	[7:0]	The cmd_buf byte 3. Mapped to command line bit [39:32] 命令 buf 字节 3，映射到命令线上 bit[39:32]	8'h00
0x50	cmd_buf4		RW	[7:0]	The cmd_buf byte 4. Mapped to command line bit [47:40] 命令 buf 字节 4，映射到命令线上 bit[47:40]	8'h00
0x54	cmd_buf5		RW	[7:0]	The cmd_buf byte 5. Mapped to command line bit [55:48] 命令 buf 字节 5，映射到命令线上 bit[55:48]	8'h00

0x58	cmd_buf6		RW	[7:0]	The cmd_buf byte 6. Mapped to command line bit [63:56] 命令 buf 字节 6，映射到命令线上 bit[63:56]	8'h00
0x5C	cmd_buf7		RW	[7:0]	The cmd_buf byte 7. Mapped to command line bit [71:64] 命令 buf 字节 7，映射到命令线上 bit[71:64]	8'h00
0x60	cmd_buf8		RW	[7:0]	The cmd_buf byte 8. Mapped to command line bit [79:72] 命令 buf 字节 8，映射到命令线上 bit[79:72]	8'h00
0x64	cmd_buf9		RW	[7:0]	The cmd_buf byte 9. Mapped to command line bit [87:80] 命令 buf 字节 9，映射到命令线上 bit[87:80]	8'h00
0x68	cmd_buf10		RW	[7:0]	The cmd_buf byte 10. Mapped to command line bit [95:88] 命令 buf 字节 10，映射到命令线上 bit[95:88]	8'h00
0x6C	cmd_buf11		RW	[7:0]	The cmd_buf byte 11. Mapped to command line bit [103:96]	8'h00

					命令 buf 字节 11，映射到命令线上 bit[103:96]	
0x70	cmd_buf12		RW	[7:0]	The cmd_buf byte 12. Mapped to command line bit [111:104] 命令 buf 字节 12，映射到命令线上 bit[111:104]	8'h00
0x74	cmd_buf13		RW	[7:0]	The cmd_buf byte 13. Mapped to command line bit [119:112] 命令 buf 字节 13，映射到命令线上 bit[119:112]	8'h00
0x78	cmd_buf14		RW	[7:0]	The cmd_buf byte 14. Mapped to command line bit [127:120] 命令 buf 字节 14，映射到命令线上 bit[127:120]	8'h00
0x7C	cmd_buf15		RW	[7:0]	The cmd_buf byte 15. Mapped to command line bit [135:128] 命令 buf 字节 15，映射到命令线上 bit[135:128]	8'h00
0x80	buf_ctrl		RW	[15]	数据缓冲清除使能 1： 触发数据缓冲清除； 0： 保持 当写 1 后，此寄存器在一个时钟后自动清	1'b0

					零；	
			RW	[14]	DMA 请求屏蔽 0： 不屏蔽； 1： 屏蔽； 注意： 请在使能 dma 之前配置此寄存器； 在使能 dma 后再配置此寄存器无效果；	1’b0
			RW	[13]	RSV	1’b0
			RW	[12]	数据 FIFO 状态信号屏蔽配置 1：激活 0：默认 数据 FIFO 状态信号，高有效； 读取卡设备时屏蔽 FIFO full 信号； 写卡设备时屏蔽 FIFO 空信号；	1’b0
			RW	[11]	设置缓冲访问方向 1： 写 0： 读	1’b0
			RW	[10]	DMA 硬件接口使能： 1： 使能 DMA 接口； 0： AHB 接口访问数据缓存； 当使用 DMA 接口时，当数据传输完成后将 自动复位此 bit（单数据块传输或者多数据	1’b0

					块传输）	
			RW	[9:2]	数据缓存数据水线设置；只有当 buf_ctl[10]=1 时有效； 注意：数据缓存深度为 128 个 word，不要配置此寄存器大于 127.	8’d0
			RO	[1]	数据缓存空信号	1’b1
			RO	[0]	数据缓存满信号	1’b0
0x100~ 0x2FF	data_buf		RW		数据缓存	NA

Bit5	Bit4	Bit3	Speed
0	0	0	1/2 base clock
0	0	1	1/4 base clock
0	1	0	1/6 base clock
0	1	1	1/8 base clock
1	0	0	1/10 base clock
1	0	1	1/12 base clock
1	1	0	1/14 base clock
1	1	1	1/16 base clock

Table 2 Mmc_ctrl[5:3] 详细定义

Note：当 mmc_ctrl[6] = 1 ，控制器工作在高速模式时，base clk = hclk；

当 mmc_ctrl[6] = 0 ，控制器工作在低速模式时，base clk = clk1m；

Clk1m= Fhclk/((mmc_cardsel[5:0]+1)*2)；

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	操作模式	传输字节
x	x	x	x	x	0	x	0	无操作	N/A
x	x	1	x	0	Trig	x	0	生成 8 null clk	N/A
0	0	0	x	0	Trig	x	0	发送命令	6
0	0	0	0	1	Trig	x	0	接收响应	6
0	0	0	1	1	Trig	x	0	接收响应	17
1	0	0	0	0	Trig	x	0	传输命令+生成 8 null clk	N/A

0	1	0	0	0	Trig	x	0	传输命令+接收响应	N/A
1	1	0	0	0	Trig	x	0	传输命令+接收响应+生成 8 null clk	N/A
x	x	x	x	x	0	1	Trig	读取单个数据+8 null clk	mmc_bytecnt
x	x	x	x	x	0	0	Trig	写单个数据+8 null clk	mmc_bytecnt

Table 3 Mmc_io[7:0] 详细定义

Note:

1. Table 3 中除了最后两行，其他操作会产生 CMD DONE 中断;

2. Table 3 中最后两行操作会产生数据传输完成中断。

Bit3	Bit2	data_crcl 和 data_crch 寄存器显示内容
0	0	DAT0 线上数据 CRC 值
0	1	DAT1 线上数据 CRC 值
1	0	DAT2 线上数据 CRC 值
1	1	DAT3 线上数据 CRC 值

Table 4 mmc_crctrl[3:2] 详细定义

Bit7	Bit6	Bit2	Bit1	Bit0	操作描述	传输字节数
mmc_io		mmc_io_mbctl				
1	1	Trig	0	0	写多块命令+响应+8 null clock +数据	mmc_blockcnt
1	1	Trig	1	0	读多块命令+响应+8 null clock +数据	mmc_blockcnt
x	x	0	0	Trig	写多块数据	mmc_blockcnt
x	x	0	1	Trig	读多块数据	mmc_blockcnt

Table 5 mmc_io[7:6] 和 mmc_io_mbctl[2:0] 详细定义

Note：

1. Table 5 中前两列操作会产生多块数据完成中断，每块数据会产生数据完成中断，以及 CMD DONE 中断；

2.当超时发生时，多块数据完成中断将不会产生，而会产生超时中断。

Bit7	Bit6	时间单位
0	0	1us
0	1	100us
1	0	10ms
1	1	1s

Table 6 mmc_io_mbctl[7:6] NAC 超时中断单位选择

Bit7	Bit6	时间单位
0	0	1us
0	1	100us
1	0	10ms
1	1	1s

Table 7 mmc_io_mbctl[5:4] 端口超时中断单位选择

Note：

1. 当使用 DMA 接口时，DMA 使能需要先打开；

2. 如果不使用中断，当传输 命令/响应/8 null clock 时，可以查询 mmc_io[2]；当需要传输数据时，可以查询 mmc_io[0]；当传输多块数据时，可以查询 mmc_io_mbctl[2] /mmc_io_mbctl0]；

3. 当 Ncr 超时发生时，数据传输会中断，控制器需要重新配置 一个新的传输；

14 SPI 控制器

14.1 功能概述

SPI 是串行外设接口（Serial Peripheral Interface）的缩写。SPI 是一种高速、全双工、同步的通信总线。SPI 的通信原理很简单，它以主从方式工作，这种模式通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以（单向传输时）,包括 SDI（数据输入）、SDO（数据输出）、SCLK（时钟）、CS（片选）。

14.2 主要特性

- 既可作为 SPI 主设备，也可作为 SPI 从设备
- 发送和接收通路各有 8 个字深度的 FIFO
- master 支持 motorola spi 的 4 种格式（CPOL，CPHA），TI 时序，macrowire 时序
- slave 支持 motorola spi 的 4 种格式（CPOL，CPHA）
- 支持全双工和半双工
- 主设备支持 bit 传输，最大支持 65535bit 传输
- 从设备支持各种长度 byte 的传输模式
- 从设备输入的 spi_clk 最大时钟频率为系统 APB 时钟的 1/6

14.3 功能描述

14.3.1 主从可配

SPI 控制器既支持设备作为 SPI 通信主设备，也支持设备作为 SPI 通信从设备。通过设置 SPI_CFG 寄存器的 Bit2 可以来回切换设备主从角色。

14.3.2 多种模式支持

作为主设备时，通过设置 SPI_CFG 寄存器的 Bit1(CPHA) 和 Bit0(CPOL)，可以使其分别以 MOTOROLA SPI 的四种格式传输数据。CPOL 是用来决定 SCK 时钟信号空闲时的电平，CPOL=0，空闲电平为低电平，CPOL=1 时，空闲电平为高电平。CPHA 是用来决定采样时刻的，CPHA=0，在每个周期的第一个时钟沿采样，CPHA=1，在每个周期的第二个时钟沿采样。还可以通过设置 TRANS_MODE 寄存器来设置主设备以 TI 时序或者 microwire 时序来传输数据，两种时序下的传输数据长度均可调。

作为从设备时，则只支持 MOTOROLA SPI 的四种格式，格式选择也是通过设置与作为主设备时相同的寄存器来实现。

14.3.3 高效的数据传输

FIFO 存储器是一个先入先出的双口缓冲器，即第一个进入其内的数据第一个被移出，其中一个存储器的输入口，另一个口是存储器的输出口。SPI 控制器集成了两个(收发各一个)深度均为 8 个字的 FIFO 存贮器，以增加数据传输率、处理大量数据流、匹配具有不同传输率的系统，从而提高了系统性能。可以通过设置 MODE_CFG 寄存器的 Bit[8:6]和 Bit[4:2]可以设置 RXFIFO 和 TXFIFO 的 trigger level，以满足不同传输速率下的性能要求。FIFO 的 trigger level 被触发后，就可以触发中断或者 DMA，将数据从内存移到 TXFIFO 或者将将数据从 RXFIFO 搬移至内存。

14.4 寄存器描述

14.4.1 寄存器列表

表 109 SPI 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	通道配置寄存器	CH_CFG	RW	用以对收发通道的相关项进行一些配置	0X0000_0000
0X0004	SPI 配置寄存器	SPI_CFG	RW	对 SPI 通信相关项进行配置	0X0000_0004
0X0008	时钟配置寄存器	Clk_CFG	RW	用以设置时钟分频系数	0X0000_0000
0X000C	模式配置寄存器	MODE_CFG	RW	配置传输模式	0X0000_0000
0X0010	中断控制寄存器	SPI_INT_MASK	RW	屏蔽或者使能相关中断	0X0000_00FF
0X0014	中断状态寄存器	SPI_INT_SOURCE	RW	用于查询中断源	0X0000_0000
0X0018	SPI 状态寄存器	SPI_STATUS	RO	列举 SPI 通信中的相关状态	0X0000_0000
0X001C	SPI 超时寄存器	SPI_TIME_OUT	RW	设置 SPI 通信超时	0X0000_0000
0X0020	数据发送寄存器	SPI_TX_DATA	RW	TX FIFO，用于存放待发送数据	0X0000_0000
0X0024	传输模式寄存器	TRANS_MODE	RW	设置传输模式	0X0000_0000
0X0028	数据长度寄存器	SLV_XMIT_LEN	RO	作为从设备时用于存放发送出去或者接收到的数据的长度	0X0000_0000
0X002C		RSV		保留	0X0000_0000
0X0030	数据接收寄存器	SPI_RX_DATA	RW/RO	RX FIFO，用于存放接收到的数据	0X0000_0000

14.4.2 通道配置寄存器

表 110 SPI 通道配置寄存器

位	访问	操作说明	复位值
[31]		RSV	1'b0
[30:23]	RW	RX_INVALID_BIT	8'h0

		表示接收通路在开始接收时，前多少 bit 是无效数据，这些无效数据需要直接扔掉，不进入 Rx FIFO。只有后续的数据进入 Rx FIFO 本寄存器与 Tx/Rx length 配合使用。最后实际存入 Rx FIFO 的数据量为 Tx/Rx length - RX_INVALID_BIT 注：master 模式有效 Motorola/ TI 模式有效	
[22]	RW	Clear FIFOs，清除 Tx 和 Rx FIFO 的内容,同时同步复位 master 和 slave 所有电路（配置寄存器除外） 1'b0：不清除 FIFO 1'b1：清除有效 软件置 1，硬件清 0 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1'b0
[21]	RW	continue mode，该模式下，spi 发送不受 Tx FIFO 空的影响。，持续传输，直至整个传输过程完成。 1'b0：normal,Tx FIFO 空后，需要等待 FIFO 中出现数据，SCK 停止翻转，同理，Rx FIFO 满后，SCK 停止翻转，等待 RX FIFO 有空间接收数据 1'b1：continue mode，Tx fifo 空，仍可以传输，直至传输完成，但此时如果 rx fifo 满，则需要暂停传输，直至 rx fifo 可以存数为止 注：master 有效 一般情况下，不设置该模式。	1'b0

		开启该模式时，如果 tx fifo 中没有数据，有可能导致无效数据先发送出去。所以请 先填入数据后，再启动 spi master Motorola/TI/microwire 模式有效	
[20]	RW	RxChOn，接收通路是否开启 1'b0: Rx channel off 1'b1: Rx channel on 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1'b0
[19]	RW	TxChOn，发送通路是否开启 1'b0: Tx channel off 1'b1: Tx channel on 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1'b0
[18:3]	RW	Tx/Rx length Spi 在传输时，有效的 SCK 数 也间接的反映了发送或者接收数据的长度。 当（TxChOn=1，RxChOn=1）时，表示发送的 bit 数，以及最大接收的 bit 数（具 体接收多少与 RX_INVALID_BIT 有关） 当（TxChOn=1，RxChOn=0）时， 表示发送的 bit 数， 当（TxChOn=0，RxChOn=1）时， 表示最大接收的 bit 数(具体接收多少与 RX_INVALID_BIT 有关,实际接收数为 Tx/Rx	16'h0

		length - RX_INVALID_BIT） 当（TxChOn=0，RxChOn=0）时， 无意义。 注：master 有效 Motorola/TI/microwire 模式有效	
[2]	RW	Chip selects 1’b0：SPI_CS 有效信号为 0 1’b1：SPI_CS 有效信号为 1 注：master 有效 Motorola/TI/microwire 模式有效	1’b0
[1]	RW	Force CS out 1’b0：spi_cs 信号输出由硬件控制 1’b1：spi_cs 信号输出由软件控制，具体输出值为 Chip selects 该信号配合 Chip selects 可以实现输出 csn 信号可编程，即该信号为 1 时，spi_cs = Chip selects 注：master 有效 Motorola/TI/microwire 模式有效	1’b0
[0]	RW	SPI start, 命令 SPI 开始接收或者发送，写 1 为 spi 开始工作，之后，自动归零 1’b0：停止 spi 工作 1’b1：启动 spi 的一次发送或者接收，自动归零 注：master 有效	1’b0

		Motorola/TI/microwire 模式有效	

14.4.3 SPI 配置寄存器

表 111 SPI 配置寄存器

位	访问	操作说明	复位值
[31:19]		RSV	13'h0
[18:17]	RW	FRAM FORMAT 2'b00:motorola 2'b01:TI 2'b10:microwire 2'b11:RSV 选择 master 支持那个厂家的协议 注：master 有效	2'b0
[16]	RW	SPI_TX pin always driven 1'b0：spi 输出只有在 spi_cs 有效时，被驱动，其它时间为三态 1'b1：spi 输出一直有驱动，即使没有数据传输 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1'b0
[15]		RSV	1'b0
[14:12]	RW	cs hold， spi_cs 在数据传输完成后持续有效的时间，即 spi_cs 的 hold 时间 3'b000 >=1 个 APB 总线 CLK 3'b001 >=2 个 APB 总线 CLK 3'b010 >=4 个 APB 总线 CLK	3'b0

		3’b011 >=8 个 APB 总线 CLK 3’b100 >=16 个 APB 总线 CLK 3’b101 >=32 个 APB 总线 CLK 3’b110 >=64 个 APB 总线 CLK 3’b111 >=127 个 APB 总线 CLK 注： master 有效 Motorola 模式有效	
[11:9]	RW	cs setup， spi_cs 在数据传输前提前有效的时间，即 spi_cs 的 setup 时间 3’b000 >=1 个 APB 总线 CLK 3’b001 >=2 个 APB 总线 CLK 3’b010 >=4 个 APB 总线 CLK 3’b011 >=8 个 APB 总线 CLK 3’b100 >=16 个 APB 总线 CLK 3’b101 >=32 个 APB 总线 CLK 3’b110 >=64 个 APB 总线 CLK 3’b111 >=127 个 APB 总线 CLK 注：master 有效 Motorola 模式有效	3’b0
[8:7]	RW	SPI-out delay, SPI 数据输出相对与 SCK 的 delay，主要是为了 hold time 考虑。 [8:7] 系统时钟周期数(APB clock) 2’b00 0 2’b01 1	2’b0

		2'b10 2 2'b11 3 注：master/slave 都有效 Motorola 模式有效	
[6:4]	RW	Frame delay，默认在一帧（spi_cs 有效期间）传输结束到下一帧开始的间隔为 SCK 时钟周期的一半，即 SPI_CS 无效时间。但为了兼容性，此处可配置。默认至少 0.5SCK [6:4] SCK clock 3'b000 0 3'b001 2 3'b010 4 3'b011 8 3'b100 16 3'b101 32 3'b110 64 3'b111 127 例如，按照 block 模式传输 128byte 的数据，数据传输完成后，会加入所设置的延迟时间。 注：master 有效	3'b0
[3]	RW	Bigendian 1'b0：数据格式采用小端模式，即传输过程中，先发低字节 1'b1：数据格式采用大端模式，即传输过程中，先发高字节	1'b0
[2]	RW	MASTER/SLAVE	1'b1

		1'b0：slave，该设备是 slave 1'b1：master，该设备是 master 注：master/slave 都有效	
[1]	RW	SPI CPHA 1'b0：传输模式 A 1'b1：传输模式 B 注：master/slave 都有效 Motorola 模式有效	1'b0
[0]	RW	SPI CPOL，SCK 在 IDLE 时的极性 1'b0：SCK IDLE 时为 0 1'b1：SCK IDLE 时为 1 注：master/slave 都有效 Motorola 模式有效	1'b0

14.4.4 时钟配置寄存器

表 112 SPI 时钟配置寄存器

位	访问	操作说明	复位值
[31:16]		RSV	16'h0
[15:0]	RW	Divider $F_{SCK} = F_{APB\_CLK} / (2 \times (Divider + 1))$ 注：master 有效 Motorola/TI/microwire 模式有效	16'h0
[31:16]		RSV	16'h0

[15:0]	RW	Divider $F_{SCK} = F_{APB\_CLK} / (2 x (Divider +1))$ 注：master 有效 Motorola/TI/microwire 模式有效	16'h0

14.4.5 模式配置寄存器

表 113 SPI 模式配置寄存器

位	访问	操作说明	复位值
[31:9]		RSV	23'h0
[8:6]	RW	RxTrigger level RX FIFO 存储的数据触发中断或者 DMA 请求的阈值：0~7word 只有 rxbuffer 中的数据大于 RxTrigger level，才会触发中断或者请求 DMA 搬移 注：master/slave 都有效 Motorola/TI/microwire 模式有效	3'b0
[5]		RSV	1'b0
[4:2]	RW	TxTrigger level TX FIFO 存储的数据触发中断或者 DMA 请求的阈值：0~7word 只有 txbuffer 中的数据大于等于 TxTrigger level，才会触发中断或者请求 DMA 搬移 注：master/slave 都有效 Motorola/TI/microwire 模式有效	3'b0
[1]	RW	RxDMA On，采用 DMA 搬移数据使能	1'b0

		1’b0：不采用 DMA， 1’b1：采用 DMA 注：master/slave 都有效 Motorola/TI/microwire 模式有效	
[0]	RW	TxDMA On，采用 DMA 搬移数据使能 1’b0：不采用 DMA， 1’b1：采用 DMA 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0

14.4.6 中断控制寄存器

表 114 SPI 中断控制寄存器

位	访问	操作说明	复位值
[31:8]		RSV	24’h0
[7]	RW	IntEn_spi_timeout 1’b0： 允许产生 spi_timeout 中断 1’b1： 不允许产生 spi_timeout 中断 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b1
[6]	RW	IntEn_spi_done 1’b0：spi 发送或者接收完成，允许产生中断 1’b1：spi 发送或者接收完成，不允许产生中断	1’b1

		注：master/slave 都有效 Motorola/TI/microwire 模式有效	
[5]	RW	IntEnRxOverrun 1’b0：Rx FIFO overflow 中断使能 1’b1：Rx FIFO overflow 中断不使能 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b1
[4]	RW	IntEnRxUnderrun 1’b0：Rx FIFO underflow 中断不使能 1’b1：Rx FIFO underflow 中断使能 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b1
[3]	RW	IntEnTxOverrun 1’b0：Tx FIFO overflow 中断使能 1’b1：Tx FIFO overflow 中断不使能 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b1
[2]	RW	IntEnTxUnderrun 1’b0：Tx FIFO underflow 中断使能 1’b1：Tx FIFO underflow 中断不使能 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b1
[1]	RW	IntEnRxFifoRdy 1’b0：Rx FIFO 有数据上传中断使能 1’b1：Rx FIFO 有数据上传中断不使能 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b1
[0]	RW	IntEnTxFifoRdy	1’b1

		1’b0：Tx FIFO 可以向 TX FIFO 写数据中断使能 1’b1：Tx FIFO 可以向 TX FIFO 写数据中断不使能 注：master/slave 都有效 Motorola/TI/microwire 模式有效	

14.4.7 中断状态寄存器

表 115 SPI 中断状态寄存器

位	访问	操作说明	复位值
[31:8]		RSV	24’h0
[7]	RW	spi_timeout 1’b0：rxfifo 中没有结尾数据需要 CPU 取走 1’b1：rxfifo 中有结尾数据需要 CPU 取走 写 1 清零 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0
[6]	RW	spi_done 1’b0：SPI 发送或者接收没有完成 1’b1：SPI 发送或者接收完成 写 1 清零 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0
[5]	RW	RxOverrun 1’b0：Rx FIFO overflow 1’b1：Rx FIFO overflow 写 1 清零	1’b0

		注：master/slave 都有效 Motorola/TI/microwire 模式有效	
[4]	RW	RxUnderrun 1’b0：Rx FIFO underflow 1’b1：Rx FIFO underflow 写 1 清零 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0
[3]	RW	TxOverrun 1’b0：Tx FIFO　overflow 1’b1：Tx FIFO　overflow 写 1 清零 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0
[2]	RW	TxUnderrun 1’b0：Tx FIFO　underflow 1’b1：Tx FIFO　underflow 写 1 清零 在 continue mode = 1 的情况下，永远不会产生该中断。 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0
[1]	RW	RxFifoRdy 1’b0：Rx FIFO 数据量<= RxTrigger level，不需要上传 1’b1：Rx FIFO 数据量> RxTrigger level，要求上传 写 1 清零 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0

[0]	RW	TxFifoRdy 1’b0：Tx FIFO 数据量 > TxTrigger level，不可以向 TX FIFO 写数据 1’b1：Tx FIFO 数据量 <= TxTrigger level，可以向 TX FIFO 写数据 写 1 清零 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0

14.4.8 SPI 状态寄存器

表 116 SPI 状态寄存器

位	访问	操作说明	复位值
[31:13]		RSV	19’h0
[12]	RO	SPI Busy 1’b0：SPI 没有发送和接收任务 1’b1：SPI 处于发送或者接收过程 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1’b0
[11:6]	RO	RX FIFO fill level Rx FIFO 中数据量，单位为字节 注：master/slave 都有效 Motorola/TI/microwire 模式有效	6’h0
[5:0]	RO	Tx FIFO fill level Tx FIFO 中数据量，单位为字节 注：master/slave 都有效 Motorola/TI/microwire 模式有效	6’h0

14.4.9 SPI 超时寄存器

表 117 SPI 超时寄存器

位	访问	操作说明	复位值
[31]	RW	spi_timer_en 1'b0：不允许 timer 计时 1'b1:允许 timer 计时 注：master/slave 都有效 Motorola/TI/microwire 模式有效	1'b0
[30:0]	RW	SPI_TIME_OUT 当一次传输完成后，在接收通路 rxfifo 中，结尾的数据如果不能触发接收中断 RxFifoRdy 或者 DMA 请求时，需要采用计时机制来通知 CPU 搬走结尾数据。 具体方法：当 rxfifo 处于 idle 状态下（没有读写操作，没有 dma 请求，cs 无效，rxfifo 中有数据，且数据量小于等于 RxTrigger level），开始计数，达到本寄存器设置的值，则触发 timeout 中断，请求 CPU 搬走结尾数据。 任何对 rxfifo 的读写操作都会清楚 timeout 计时器。 所表示的时间为： $T = SPI\_TIME\_OUT/F_{APB\_CLK}$ 注：master/slave 都有效 Motorola/TI/microwire 模式有效	31'h0

14.4.10 数据发送寄存器

表 118 SPI 数据发送寄存器

位	访问	操作说明	复位值

[31:0]	RW	向 Tx FIFO 写数据的窗口地址 注： master/slave 都有效 Motorola/TI/microwire 模式有效	32'h0

14.4.11传输模式寄存器

表 119 SPI 传输模式寄存器

位	访问	操作说明	复位值
[31:30]		RSV	16'd0
[29:24]	RW	TI_BLK_LEN 在 TI 的时序模式下，每个 block 传输的长度，即每次 CS 有效之后的传输数据长度。 支持 4~32bit 6'h4: 4bit 长数据 6'h5: 5bit 长数据 6'h6: 6bit 长数据 …….. 6'h20: 32bit 长数据 注：master 有效 TI 模式有效	6'd0
[16]	RW	MICRO_BURST 1b'1：Microwire 模式下，采用 burst 传输，即 Tx 发送控制字，Rx 接收数据，依次交替进行，MICRO_CONTROL_LEN 表示的是控制字长度，MICRO_DAT_LEN 表示的	1'b0

		是发送或者接收字的长度，Tx/Rx length 表示的是整个传输过程中有效 sck，burst 模式下，发送接收交替进行的次数为（Tx/Rx length）/(MICRO_CONTROL_LEN+MICRO_DAT_LEN+1) 1'b0：Microwire 模式下，不采用 burst 传输 在此模式下，有两种情况 1）tx_ch_on = 1, rx_ch_on = 0,此时，只有发送，MICRO_CONTROL_LEN 表示的是控制字长度，Tx/Rx length 表示的是整个传输过程中有效 sck，此时发送的数据长度为 m*MICRO_DAT_LEN = Tx/Rx length - MICRO_CONTROL_LEN，其中 m 表示发送多少个（MICRO_DAT_LEN）长度的字 2）tx_ch_on = 1, rx_ch_on = 1，此时，Tx 发送控制字，Rx 接收数据，MICRO_CONTROL_LEN 表示的是控制字长度，Tx/Rx length 表示的是整个传输过程中有效 sck，接收数据长度为 m*MICRO_DAT_LEN = Tx/Rx length-MICRO_CONTROL_LEN-1，其中 m 表示接收多少个（MICRO_DAT_LEN）长度的字 注：master 有效 microwire 模式有效	
[13:8]	RW	MICRO_DAT_LEN Microwire 模式下，在 burst 模式模式时，每个 burst 传输数据的长度 从 1~32： 6'h1: 1bit 长数据 6'h2: 2bit 长数据	6'd0

		6’h3:　3bit 长数据 …….. 6’h20　32bit 长数据 注：master 有效 microwire 模式有效	
[5:0]	RW	MICRO_CONTROL_LEN Microwire 模式下，命令字的长度 从 1~32： 6’h1:　1bit 长命令 6’h2:　2bit 长命令 6’h3:　3bit 长命令 …….. 6’h20　32bit 长命令 注：master 有效 microwire 模式有效	6’d0

14.4.12 数据长度寄存器

表 120 SPI 数据长度寄存器

位	访问	操作说明	复位值
[31:16]	RO	作为 slave 时，在 cs 有效期间，发送出去的数据长度，单位为 bit 注：slave 有效 Motorola 模式有效	16’h0

[15:0]	RO	作为 slave 时，在 cs 有效期间，接收到的数据长度，单位为 bit 注：slave 有效 Motorola 模式有效	16'h0

14.4.13 数据接收寄存器

表 121 SPI 数据接收寄存器

位	访问	操作说明	复位值
[31:0]	RO	从 Rx FIFO 读数据的窗口地址 注：master/slave 都有效 Motorola/TI/microwire 模式有效	32'h0

15 I2C 控制器

15.1 功能概述

I2C 总线是一种简单、双向二线制同步串行总线。它只需要两根线即可在连接于总线上的器件之间传送信息。

主器件用于启动总线传送数据，并产生时钟以开放传送的器件，此时任何被寻址的器件均被认为是从器件．在总线上主和从、发和收的关系不是恒定的，而取决于此时数据传送方向。如果主机要发送数据给从器件，则主机首先寻址从器件，然后主动发送数据至从器件，最后由主机终止数据传送；如果主机要接收从器件的数据，首先由主器件寻址从器件．然后主机接收从器件发送的数据，最后由主机终止接收过程。在这种情况下．主机负责产生定时时钟和终止数据传送。

15.2 主要特性

- APB 总线协议标准接口
- 只可作为主设备控制器使用
- I2C 工作速率可配，100KHz~400KHz
- 多路 GPIO 可复用成 I2C 的通信接口
- 可快速输出和检测时序信号

15.3 功能描述

15.3.1 传输速率选择

通过设置寄存器 PRERlo 和寄存器 PRERhi 就可以将 I2C 总线上的数据传输速率配置在 100KHz 到

400KHz 之间的任意总线频率整数分频值。

15.3.2 中断及启动停止可控

通过设置寄存器 CTR 的 Bit6 允许或者禁止 I2C 控制器产生中断，并且还可以通过设置 Bit7 来随时启动或者停止 I2C 控制器的工作。

15.3.3 快速输出及检测信号

通过设置寄存器 CR_SR 的相应位可以使控制器快速输出或者检测总线 START 信号，总线 STOP 信号，总线 ACK 信号，总线 NACK 信号。在主模式下，I2C 接口启动数据传输并生成时钟信号。 一个串行数据传输始终以启动信号开始，以停止信号结束。一旦在总线上生成启动信号，就选择了主设备模式。

15.4 寄存器描述

15.4.1 寄存器列表

表 122 I2C 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	时钟分频寄存器_1	PRERlo	RW	存放低 8 位的分频值，用以对 APB 总线时钟进行分频	0X0000_00FF
0X0004	时钟分频寄存器_2	PRERhi	RW	存放高 8 位的分频值，用以对 APB 总线时钟进行分频	0X0000_00FF
0X0008	控制寄存器	CTR	RW	用以控制中断的使能以及 I2S 控制器的使能	0X0000_0040
0X000C	数据寄存器	TXR_RXR	RW	用以存放待发送的数据或者接收到	0X0000_0000

				的数据	
0X0010	收发控制寄存器	CR_SR	RW	用以控制一些数据读写相关的操作	0X0000_0000
0X0014	TXR 读出寄存器	TXR	RO	读取 I2C 发送时的 TXR 寄存器值	0X0000_0000
0X0018	CR 读出寄存器	CR	RO	读取 I2C 控制寄存器的设定值 CR	0X0000_0000

15.4.2 时钟分频寄存器_1

表 123 I2C 时钟分频寄存器_1

位	访问	操作说明	复位值
[31: 8]		保留	
[7 : 0]	RW	时钟分频配置 prescale 的低 8bit。 例如： apb_clk=40MHz, SCL=100KHz prescale = (40*1000)/(5*100) - 1 = 16'd79 apb_clk = 40M, SCL=400K prescale=(40*1000)/(5*400) - 1 = 16'd19	8'hff

15.4.3 时钟分频寄存器_2

表 124 I2C 时钟分频寄存器_2

位	访问	操作说明	复位值
[31: 8]		保留	
[7 : 0]	RW	时钟分频配置 prescale 的高 8bit。 例如：	8'hff

		apb_clk=40MHz, SCL=100KHz prescale = (40*1000)/(5*100) - 1 = 16'd79 apb_clk = 40M, SCL=400K prescale=(40*1000)/(5*400) - 1 = 16'd19	

15.4.4 控制寄存器

表 125 I2C 控制寄存器

位	访问	操作说明	复位值
[31:8]		保留	
[7]	RW	I2C 使能控制， 1'b0：不使能 1'b1：使能	1'b0
[6]	RW	中断 MASK， 1'b0：允许中断产生 1'b1：不允许中断产生	1'b1
[5:0]		保留	

15.4.5 数据寄存器

表 126 I2C 数据寄存器

位	访问	操作说明	复位值
[31:8]		保留	
[7:0]	WR	写该寄存器时，为发送寄存器 TXR，	8'h0

		表示下一个要发送的字节， 当为设备地址时， [0]：1 时表示读，0 时表示写。 读该寄存器时，为接收寄存器 RXR， 为最新从 I2C 上接收到的字节。	

15.4.6 收发控制寄存器

表 127 I2C 收发控制寄存器

位	访问	操作说明	复位值
[31:8]		保留	
[7:0]	WR	写该寄存器时，为 CR，功能如下： [7]：STA，控制产生 START 时序； 1’b0：无效 1’b1：产生 START 时序 [6]：STO，控制产生 STOP 时序； 1’b0：无效 1’b1：产生 STOP 时序 [5]：RD，从 SLAVE 读； 1’b0：无效	8’h0

		1’b1：从 SLAVE 读 [4]：WR，向 SLAVE 写； 1’b0：无效 1’b1：向 SLAVE 写 [3]：控制向 SLAVE 送回 ACK/NACK； 1’b0：回 ACK 1’b1：回 NAK [2:1]：保留； [0]：IACK，清除中断状态，1 有效； 1’b0：无效 1’b1：清除中断标志 读该寄存器时，为 SR，功能如下： [7]：RxACK，从 SLAVE 收到的 ACK/NACK 状态； 1’b0：从 SLAVE 收到 ACK 1’b1：从 SLAVE 收到 NAK [6]：BUSY； 1’b0：STO 后置 0	

		1’b1：STA 后置 1 [5]：AL，Arbitration Lost，此位保留； [4:2]：保留； [1]：TIP； 1’b0：无传输正在进行 1’b1：有传输正在进行 [0]：IF，中断状态位； 1’b0：无中断产生 1’b1：传输完成或者 AL 时置 1	

15.4.7 TXR 读出寄存器

表 128 I2C TXR 读出寄存器

位	访问	操作说明	复位值
[31:8]		保留	
[7:0]	RO	只读，TXR 寄存器的读出值， 功能描述见 TXR_RXR 寄存器；	8’h0

15.4.8 CR 读出寄存器

表 129 I2C CR 读出寄存器

位	访问	操作说明	复位值

[31:8]		保留	
[7:0]	RO	只读，CR 寄存器的读出值， 功能描述见 CR_SR 寄存器；	8'h0

16 I2S 控制器

16.1 功能概述

I2S（Inter-IC Sound）是针对数字音频设备（如 CD 播放器、数码音效处理器、数字电视音响系统）之间的音频数据传输而制定的一种总线标准。它采用了独立的导线传输时钟与数据信号的设计，通过将数据和时钟信号分离，避免了因时差诱发的失真，为用户节省了购买抵抗音频抖动的专业设备的费用。标准的 I2S 总线电缆是由 3 根串行导线组成的：1 根是时分多路复用（简称 TDM）数据线；1 根是字选择线；1 根是时钟线。

16.2 主要特性

- 实现 I2S 接口，支持 I2S 和 PCM 协议
- 支持 amba APB 总线接口，32bit single 读写操作
- 支持主从模式
- 支持 8，16，24，32 位宽，最高采样频率为 192KHz
- 支持单声道和立体声模式
- 兼容 I2S 和 MSB justified 数据格式，兼容 PCM A/B 格式
- 支持 DMA 请求读写操作，只支持按字操作

16.3 功能描述

16.3.1 多种模式支持

通过设置 I2S Control 寄存器的 Bit[25:24]可以设置数据格式为 I2S 格式，MSB justified 格式，PCM A 格式，或者 PCM B 格式；通过设置 I2S Control 寄存器的 Bit[22]可以选择单声道或者立体声模式。通过

设置 I2S Control 寄存器的 Bit[5:4]可以设置数据传输字的位宽，可以设置为 8bit，16bit，24bit，32bit。

16.3.2 零交叉检测

通过设置 I2S Control 寄存器的 Bit[17:16]可以设置是否启用左右声道的零交叉检测功能；通过设置 I2S_IMASK 寄存器的 Bit[9:8]可以设置左右声道零交叉检测功能是否产生中断。若启用了检测功能，并且使了中断，则当检测到零交叉现象时，程序会执行中断子程序，同时 I2S_INT_FLAG 寄存器的 Bit[9:8]的相应位会被置 1。

16.3.3 高效的数据传输

FIFO 存储器是一个先入先出的双口缓冲器，即第一个进入其内的数据第一个被移出，其中一个存储器的输入口，另一个口是存储器的输出口。I2S 控制器集成了两个(收发各一个)深度均为 8 个字的 FIFO 存贮器，以增加数据传输率、处理大量数据流、匹配具有不同传输率的系统，从而提高了系统性能。可以通过设置 I2S Control 寄存器的 Bit[14:12]和 Bit[11:9]可以设置 RXFIFO 和 TXFIFO 的 trigger level，以满足不同传输速率下的性能要求。FIFO 的 trigger level 被触发后，就可以触发中断或者 DMA，将数据从内存移到 TXFIFO 或者将将数据从 RXFIFO 搬移至内存。

16.4 I2S/PCM 时序图

本模块提供对 4 种协议的支持，标准 I2S，MSB Justified，PCM-A,PCM-B. 通过配置寄存器 0x00[25:24] 来选择使用哪种协议。具体每种协议的接口时序见 Fig1 至 Fig4.

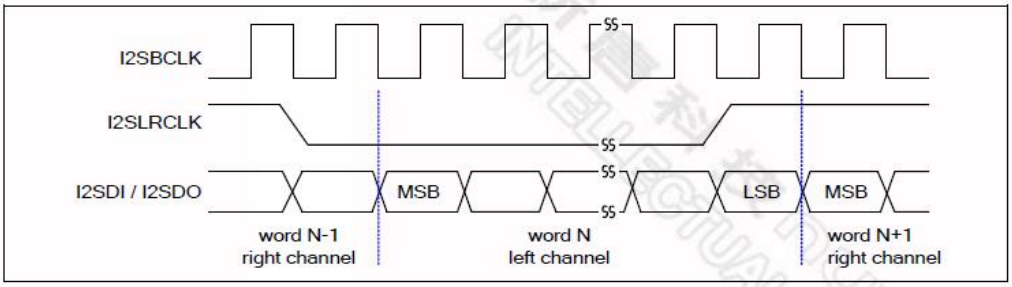


Fig1. I2S Bus Timing Diagram（PCM=0，Format=0）

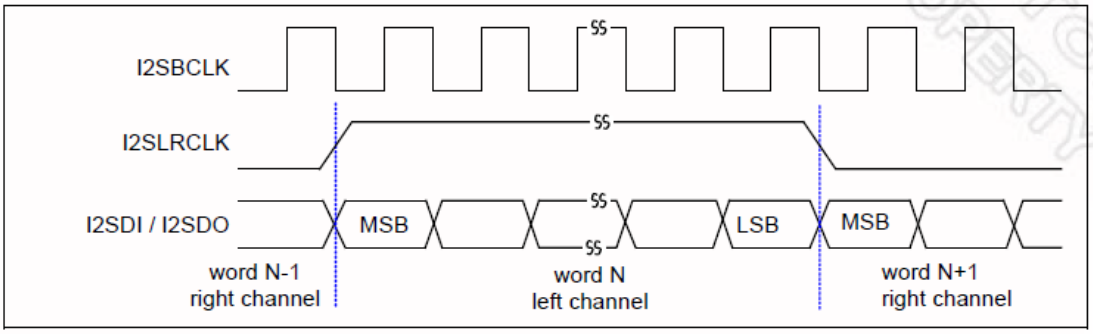


Fig2. MSB Justified Timing Diagram（PCM=0，Format=1）

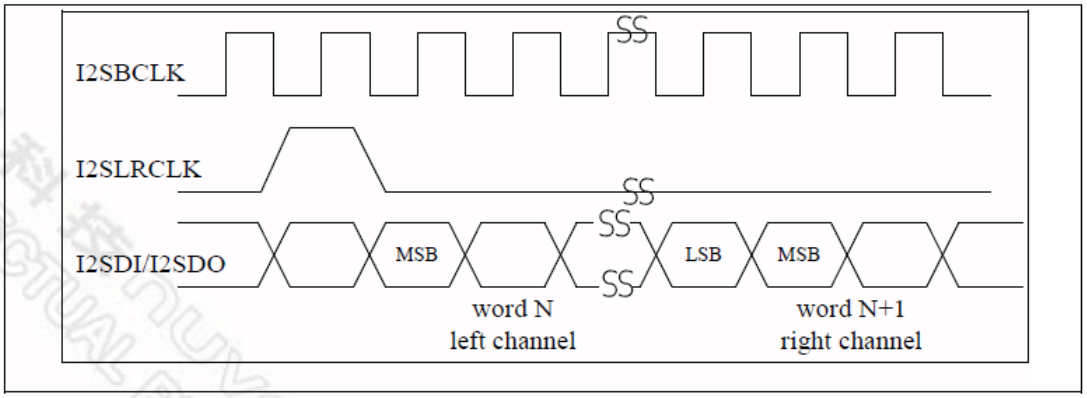


Fig3. PCM A Audio Diagram（PCM=1，Format=0）

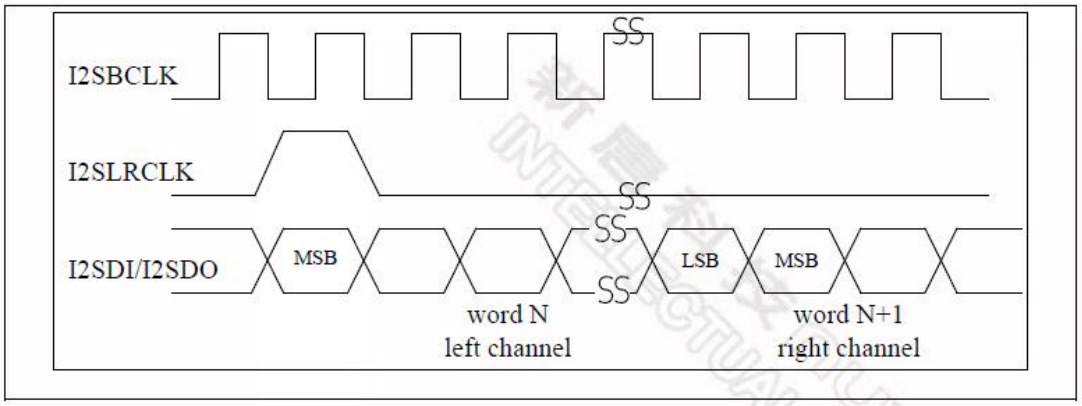


Fig4. PCM B Audio Diagram（PCM=1，Format=1）

16.5 FIFO 存储结构图

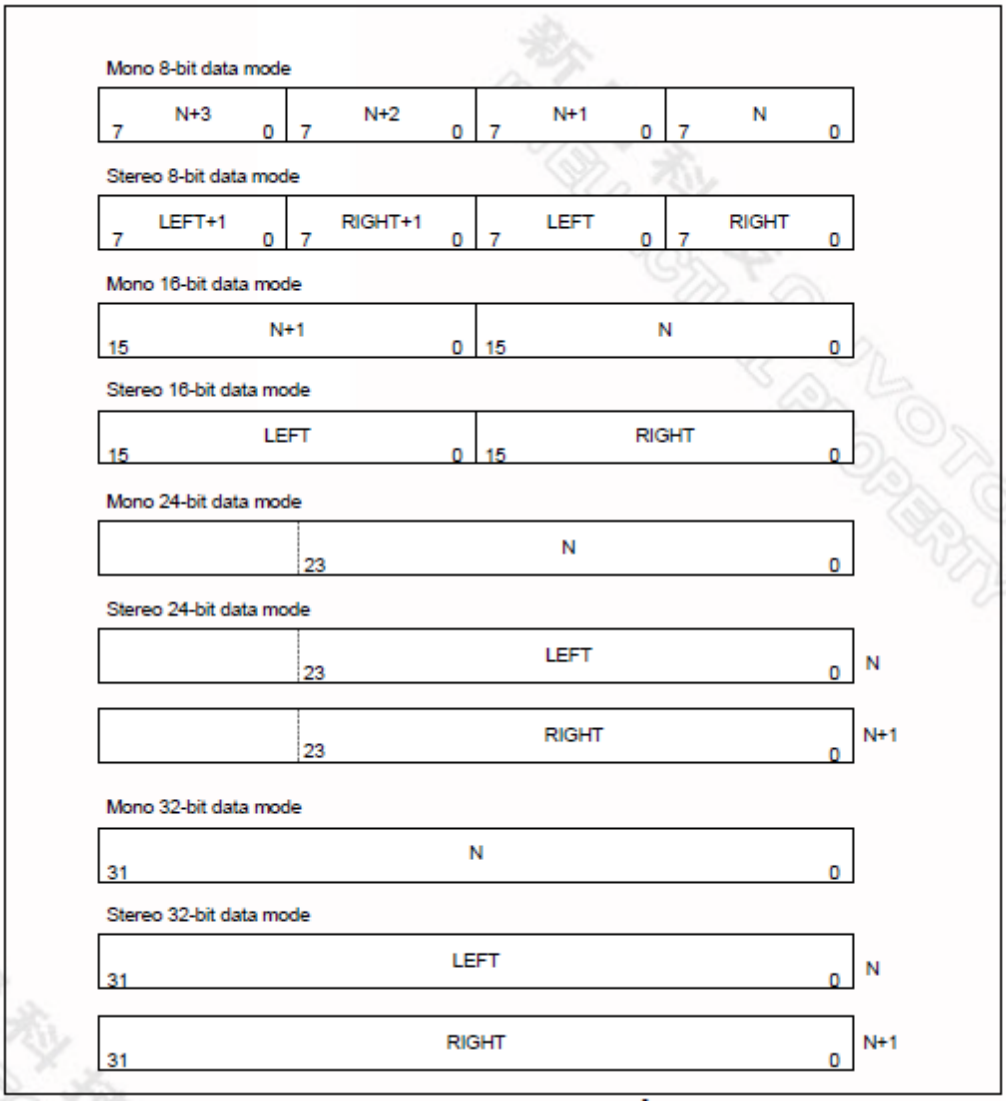


Fig 5. FIFO 存储结构

I2S 模块提供 2 个 8x32bit 大小的 FIFO，一个作为 TX FIFO，一个作为 RX FIFO。

FIFO 内数据的存储结构根据工作模式的不同有所不同。当工作在单声道 8bit 模式时，FIFO 内每个 word 可以存储 4 个 sample 数据。当工作在双声道 8bit 模式时，左右声道数据交替存储于 FIFO 内，处于低位的 byte 存储右声道数据，处于高位的 byte 存储左声道数据。一个 word 可以存储 2 个 sample 数据。单工 16bit 模式时，一个 word 可以存储 2 个 sample，双工 16bit 模式时，一个 word 存储一个 sample 数据，低 16 位为右声道，高 16 位为左声道。24bit 与 32bit 模式时每个 word 只能存储一个 sample 数据或者一个声道的数据，具体存储方式见图 5 所示。

16.6 I2S 模块工作时钟配置

I2S 模块的工作时钟配置通过设置在时钟与复位模块中 0x40000718 寄存器可以配置时钟源选用外部时钟还是内部 160MHz 时钟，是否开启 MCLK 时钟，以及 MCLK 和 BCLK 的工作频率。

通过设置寄存器 0x40000718[0]可以选择使用内部 160MHz 时钟还是使用外部时钟作为 I2S 模块时钟源。如果选用外部时钟，I2S 模块将使用从 IO- I2S_M_EXTCLK(PA_5,Option 4)输入的时钟作为模块时钟源。

而 0x40000718[1] 选择是否开启 MCLK 时钟。[7:2] 位为配置 MCLK 分频比的区域。计算公式如下：

F_mclk=F_I2SCLK/MCLKDIV

F_mclk 为实际 MCLK 频率；

F_I2SCLK 为 I2S 模块的时钟源。如果选用内部时钟，则 F_I2SCLK =160MHz；如果选用外部时钟，则 F_I2SCLK 等于外部输入的时钟频率；

MCLKDIV 即为寄存器 0x40000718[7:2] 配置的时钟分频比。需要注意的是 MCLKDIV>=2;

寄存器 0x40000718 的[17:8]位 为配置 BLCK 时钟的分频比的区域。分频比计算公式如下：

F_BCLK=F_I2SCLK/BCLKDIV

F_BCLK 为 BCLK 实际工作的频率；

F_I2SCLK 为 I2S 模块的时钟源。如果选用内部时钟，则 F_I2SCLK =160MHz；如果选用外部时钟，则 F_I2SCLK 等于外部输入的时钟频率；

BCLKDIV 即是需要配置的分频比。根据不同的工作模式需要选择不同的分频比。选择分频比的公式如下：

BCLKDIV=round (F_I2SCLK/(Fs*W*F))

Fs 为 I2S 接口上音频数据的采样频率，最高支持 192KHz；

W 为采样位宽，8/16/24/32 bit 可以选择；

F 为单声道/双声道选择。传输数据为单声道时 F=1，传输数据为双声道时 F=2；

最后计算出来的分频比进行四舍五入。

以下为根据实际工作模式选取 BCLKDIV 的例子：

若选取内部时钟为模块时钟源，需要传输 128KHz 采样率，24bit 双声道数据，则计算需要设置 BCLKDIV 的过程如下：

BCLKDIV=round(160*10e6/128*10e3*24*2)=10'd26；

分频寄存器说明如下：

0x18	I2S_Clk_Ctrl	[0]	RW	EXTAL_EN External clock select Select whether use External or Internal clock for I2S block 0=internal clk 1=external clk Note: When External clock is enabled, the external clk must be 2*N*256 fs, where fs is sample frequency and N must be integer.	1'b0

		[1]	RW	MCLKEN MCLK enable 0=MCLK disabled 1=MCLK enabled	1'b0
		[7:2]	RW	MCLKDIV MCLK divider If external clock is selected, this divider is used to produce proper MCLK frequency. F_mclk=F_I2SCLK/MCLKDIV Where MCLKDIV >=2 F_mclk=F_I2SCLK when MCLKDIV=0 Where F_I2SCLK is external clk. Note: F_mclk should be configured as 256 * fs where fs is sample frequency.	6'd0
		[17:8]	RW	BCLKDIV BCLK divider F_BCLK=F_I2SCLK/BCLKDIV Note: When EXTAL_EN is not selected, Internal PLL is used and F_I2SCLK =160MHz	10'd0

				When WXTAL_EN is enabled, F_I2SCLK = External crystal frequency **BCLKDIV=round (F_I2SCLK/(Fs*W*F))** Where Fs is sample frequency of audio data and W is word width. F=2 when data is stereo and F=1 when data is mono. For example, if internal PLL is used and the data width is 24bit, Format is stereo format, sample frequency is 128KHz. Then the BCLKDIV should be configured as (160*10e6/128*10e3*24*2) =10'd26	
		[31:18]	RO	RSV	14'd0

16.7 其他功能说明：

16.7.1 过零检测：

为了避免数据损坏导致频率突然变化而产生的噪声，I2S 模块针对每个声道都提供了过零检测的功能。

当发送的相邻两个数据符号位出现变化时，模块会产生中断，提醒 MCU 进行检测，处理；同时将后一个

数据强制静音。

16.7.2 静音功能

当静音功能打开时，数据仍将发送，但输出的数据将被强制置为 0；

16.7.3 中断

本模块提供发送/接收完成中断，左右声道过零检测中断，发送/接收 FIFO 的阈值中断（发送 FIFO 中数据低于阈值，接收 FIFO 中数据高于阈值），发送/接收 FIFO 的 underflow/overflow 中断。中断状态在寄存器 0x08 中查询。

16.7.4 FIFO 状态查询

寄存器 0x10 提供 CPU 对 I2S 模块中发送/接收 FIFO 的状态查询功能。通过寄存器 CPU 能够检查 FIFO 中剩余多少数据未处理。在接收 FIFO 中最后一个 word 中有几个 byte 是有效数据。

16.8 数据传输流程

16.8.1 主端发送音频数据

1. 配置好使用的引脚, SDO, BCLK, LRCLK
2. 参考 Section 2.4 设置时钟与复位模块中寄存器 0x40000718，配置好工作时钟频率；
3. 设置 I2S 寄存器 0x00，设置为 master mode， 配置传输格式，声道选择，数据位宽，左右声道是否开启过零检测，并设置发射通路-txen 开启。
4. 设置寄存器 0x4，使能想要使用的中断；
5. 如果使用 DMA 传输数据，在 DMA 模块中选择使用的通道，配置好发射数据的地址，长度。并在 I2S 模块寄存器中 0x0 中使能 DMA，并设置 FIFO 阈值。当 FIFO 中数据低于阈值范围即会自动请求 DMA 搬运数据。
6. 向发射 FIFO 地址-寄存器 0x10 写入需要发射的数据，使能寄存器 0x0[0]，模块即会自动从 FIFO 中取出数据送到 I2S 总线上。
7. 当 FIFO 中数据小于设置的阈值时，模块将向 DMA 模块请求数据，或者发送 TXTHIF 中断。
8. 当 FIFO 中数据全部取出后，TXDONE 中断将置位，当最后一帧发送完成，TXUDIF 中断将置位，通知 CPU 发送完成，模块停止发送。

16.8.2 从端接收音频数据

1. 配置好使用的引脚, SDI, BCLK, LRCLK
2. 设置 I2S 寄存器 0x00，设置为 slave mode 配置传输格式，声道选择，数据位宽，左右声道是否开启过零检测，并设置接收通路-rxen 开启。
3. 设置寄存器 0x4，使能想要使用的中断；
4. 如果使用 DMA 传输数据，在 DMA 模块中选择使用的通道，配置好发射数据的地址，长度。并在 I2S

模块寄存器中 0x0 中使能 DMA，并设置 FIFO 阈值。当数据高于阈值范围即会自动请求 DMA 搬运数据。

5. 使能寄存器 0x0[0]，模块检测到有效的 BCLK 和 LRCLK 后即会自动从 SDI 上采集数据存储在 FIFO 中。当 FIFO 中数据高于设置的阈值时，模块将向 DMA 请求搬运数据到内存中，或者发送 RXTHIF 中断。当 CPU 关闭 RXEN 时 RXDONE 中断产生。CPU 可以通过寄存器 0x10 查询接收 FIFO 中有多少个数据，最后一个 word 中有多少个 byte 的有效数据。然后根据 FIFO 状态处理最后剩余的数据。

16.8.3 主端接收音频数据

1. 配置好使用的硬件，SDI , SCLK, LRCLK
2. 参考 Section 2.4 设置时钟与复位模块中寄存器 0x40000718，配置好工作时钟频率；
3. 设置 I2S 寄存器 0x00，设为 master mode，配置传输格式，声道选择，数据位宽，左右声道是否开启过零检测，并设置接收通路 rxen 开启。
4. 设置寄存器 0x4，使能想要使用的中断；
5. 如果使用 DMA 传输数据，在 DMA 模块中选择使用的通道，配置好发射数据的地址，长度。并在 I2S 模块寄存器中 0x0 中使能 DMA，并设置 FIFO 阈值。当数据高于阈值范围即会自动请求 DMA 搬运数据。
6. 使能寄存器 0x0[0]，模块即会自动发送 BCLK 和 LRCLK，同时从 SDI 上采集数据存储在 FIFO 中。当 FIFO 中数据高于设置的阈值时，模块将向 DMA 请求搬运数据到内存中，或者发送 RXTHIF 中断。当 CPU 关闭 RXEN 时 RXDONE 中断产生。CPU 可以通过寄存器 0x10 查询接收 FIFO 中有多少个数据，最后一个 word 中有多少个 byte 的有效数据。然后根据 FIFO 状态处理最后剩余的数据。
7. 数据接收完成后关闭 i2s 使能。

16.8.4 从端发送音频数据

1. 配置好使用的引脚, SDO, BCLK, LRCLK
2. 设置 I2S 寄存器 0x00，设置为 slave mode， 配置传输格式，声道选择，数据位宽，左右声道是否开启过零检测，并设置发射通路 txen 开启。
3. 设置寄存器 0x4，使能想要使用的中断；
4. 如果使用 DMA 传输数据，在 DMA 模块中选择使用的通道，配置好发射数据的地址，长度。并在 I2S 模块寄存器中 0x0 中使能 DMA，并设置 FIFO 阈值。当 FIFO 中数据低于阈值范围即会自动请求 DMA 搬运数据。
5. 向发射 FIFO 地址-寄存器 0x10 写入需要发射的数据，使能寄存器 0x0[0]，当模块检测到有效的 BCLK 和 LRCLK 时，模块即会自动从 FIFO 中取出数据送到 SDO 上。
6. 当 FIFO 中数据小于设置的阈值时，模块将向 DMA 模块请求数据，或者发送 TXTHIF 中断。
7. 当 FIFO 中数据全部取出后，TXDONE 中断将置位，当最后一帧发送完成，TXUDIF 中断将置位，通知 CPU 发送完成，模块停止发送。

16.8.5 全双工模式

1. 配置好使用的引脚,SDI, SDO, BCLK, LRCLK
2. 如果是 master 模式，需要配置时钟分频。
3. 设置 I2S 寄存器 0x00，配置工作模式（主/从）， 配置传输格式，声道选择，数据位宽，左右声道是否开启过零检测，并设置发射通路 txen 和接收通路 rxen 开启。
4. 设置寄存器 0x4，使能想要使用的中断；
5. 如果使用 DMA 传输数据，在 DMA 模块中选择使用的通道，配置好发射数据的地址，长度。并在 I2S 模块寄存器中 0x0 中使能 DMA，并设置 FIFO 阈值。当 FIFO 中数据低于阈值范围即会自动请求 DMA 搬运数据。

6. 向发射 FIFO 地址-寄存器 0x10 写入需要发射的数据
7. 如果是 master 模式，使能寄存器 0x0[0]，模块会开始发送 BCLK 和 LRCLK，从发送 FIFO 中取出数据开始从 SDO 端口送出，同时从 SDI 接收数据存入接收 FIFO。
8. 如果是 slave 模式，当模块检测到有效的 BCLK 和 LRCLK 时，模块即会自动从发送 FIFO 中取出数据送到 SDO 上，同时从 SDI 接收数据存入接收 FIFO。
9. 当发送 FIFO 中数据小于设置的阈值时，模块将向 DMA 模块请求数据，或者发送 TXTHIF 中断。
10. 当 FIFO 中数据高于设置的阈值时，模块将向 DMA 请求搬运数据到内存中，或者发送 RXTHIF 中断。
11. 当 TXFIFO 中数据全部取出后，TXDONE 中断将置位，当最后一帧发送完成，TXUDIF 中断将置位，通知 CPU 发送完成，模块停止发送。
12. 当 CPU 关闭 RXEN 时 RXDONE 中断产生。CPU 可以通过寄存器 0x10 查询接收 FIFO 中有多少个数据，最后一个 word 中有多少个 byte 的有效数据。然后根据 FIFO 状态处理最后剩余的数据。

16.9 寄存器描述

16.9.1 寄存器列表

表 130 I2S 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	控制寄存器	I2S Control	RW/RO	控制 I2S 相关功能，详见下述章节；	0X0000_4800
0X0004	中断屏蔽寄存器	I2S_IMASK	RW	控制开启或关闭 I2S 中所有的中断	0X0000_03FF
0X0008	中断标志寄存器	I2S_INT_FLAG	RW/RO	中断标志位，可用于查询中断是否产生及清除相关中断	0X0000_0000
0X000c	状态寄存器	I2S_STATUS	RO	用于查询 I2S 通信过程中 FIFO 的相关状态	0X0000_0000

0X0010	数据发送寄存器	I2S_TX	WO	控制器会将它里面的数据发送到总线上	0X0000_0000
0X0014	数据接收寄存器	I2S_RX	RO	控制器会将总线上的数据接收到它里面	0X0000_0000

16.9.2 控制寄存器

表 131 I2S 控制寄存器

位	访问	操作说明	复位值
[31:29]	RO	RSV	7'h0
[28]	RW	模式选择 0 = master 模式 1 = slave 模式	1'b0
[27]	RW	双工模式选择 1 = 使能双工模式 0 = 关闭双工模式	1'b0
[26]	RW	超时计数控制位，当此位被置 1 并且传输进程被主设备强制停止时，将不发生接收完成(RXDONE)中断	1'h0
[25:24]	RW	FORMAT 数据格式选择 2'b00：I2S 数据格式 2'b01：MSB Justified 数据格式 2'b10：PCM A 声音数据格式 2'b11：PCM B 声音数据格式	2'b0
[23]	RW	RXLCH	1'b0

		声道接收使能控制位 1'b0：使能接收右声道数据 1'b1：使能接收左声道数据 注意：只有选择 MONO_STEREO 的单声道模式时，此位才有效	
[22]	RW	MONO_STEREO 单声道立体声选择位 1'b0：数据以立体声格式传输 1'b1：数据以单声道格式传输	1'b0
[21]	RW	RXDMAEN 接收 DMA 请求使能位 1'b0：不使能发送 DMA 请求 1'b1：使能发送 DMA 请求 注意:当使能传输 DMA 请求并且 RXFIFO 中的字的个数等于或者大于 RXTH 时，I2S 控制器会向 DMA 发出传输请求直至 RXFIFO 为空才停止 DMA 传输。	1'b0
[20]	RW	TXDMAEN 发送 DMA 请求使能位 1'b0：不使能发送 DMA 请求 1'b1：使能发送 DMA 请求 注意：当使能传输 DMA 请求并且 TXFIFO 中的字的个数小于 TXTH 时，I2S 控制器会向 DMA 发出传输请求直至 TXFIFO 满才停止 DMA 传输。	1'b0
[19]	WO	RXCLR 清空 RXFIFO	1'b0

		1’b0：无效 1’b1：清空 RXFIFO 注意：写 1 清空 RXFIFO，由硬件自动清空。读此位永远返回 0	
[18]	WO	TXCLR 清空 TXFIFO 1’b0：无效 1’b1：清空 TXFIFO 注意：写 1 清空 TXFIFO，由硬件自动清空。读此位永远返回 0	1’b0
[17]	RW	LZCEN 左声道零交叉检测使能控制位 1’b0：停止左声道零交叉检测 1’b1：使能左声道零交叉检测	1’b0
[16]	RW	RZCEN 右声道零交叉检测使能控制位 1’b0：停止右声道零交叉检测 1’b1：使能右声道零交叉检测	1’b0
[15]	RW	Rx_clk_phase_sel 接收时钟相位选择 1’b0：默认模式 以上面提及的 I2S 总线时序展示 1’b1：反转模式 以上面提及的 I2S 总线时序的反转形式展示	1’b0

[14:12]	RW	RXTH RXFIFO 阈值 3’b000：设置阈值为 0 个字 3’b000：设置阈值为 1 个字 … 3’b111：设置阈值为 7 个字 注意：当 RXFIFO 中现有的字等于或者多于 RXTH 的值时，RXTHIF 位会被置位。此时可以根据设置来选择触发 RXDMA 或者 I2S 中断	3’h4
[11:9]	RW	TXTH TXFIFO 阈值 3’b000：设置阈值为 0 个字 3’b000：设置阈值为 1 个字 … 3’b111：设置阈值为 7 个字 注意：当 TXFIFO 中现有的字等于或者少于 TXTH 的值时，TXTHIF 位会被置位。此时可以根据设置来选择触发 TXDMA 或者 I2S 中断	3’h4
[8]	RW	Tx_clk_phase_sel 选择发射时钟相位模式 1’b0：默认模式 以上面提及的 I2S 总线时序展示 1’b1：反转模式 以上面提及的 I2S 总线时序的反转形式展示	1’b0

[7:6]	RW	RSV	2'h0
[5:4]	RW	WDWIDTH 传输字长设置位 2'b00：字长 8 bit 2'b01：字长 16 bit 2'b10：字长 24 bit 2'b11：字长 32 bit	2'b0
[3]	RW	MUTE 传输哑声使能标志位 1'b0：从移位寄存器传输数据，正常操作模式 1'b1：将传输数据置 0，使声音静音	1'b0
[2]	RW	RXEN 接收使能标志位 1'b0：停止 I2S 数据接收 1'b1：使能 I2S 数据接收	1'b0
[1]	RW	TXEN 传输使能标志位 1'b0：停止 I2S 数据传输 1'b1：使能 I2S 数据传输	1'b0
[0]	RW	I2SEN I2S 使能标志位 1'b0：不使能	1'b0

		1’b1：使能	

16.9.3 中断屏蔽寄存器

表 132 I2S 中断屏蔽寄存器

位	访问	操作说明	复位值
[31:10]	RO	RSV	22’h0
[9]	RW	LZCIMASK 左声道零交叉中断使能标志位 1’b0：中断不使能 1’b1：中断使能 当使能中断，并且检测到左声道上有零交叉时，产生中断	1’b1
[8]	RW	RZCIMASK 右声道零交叉中断使能标志位 1’b0：中断不使能 1’b1：中断使能 当使能中断，并且检测到右声道上有零交叉时，产生中断	1’b1
[7]	RW	TXDONEMASK 发送完成中断使能位 1’b0：中断不使能 1’b1：中断使能 当使能中断，并且 TXFIFO 为空时，产生中断	1’b1
[6]	RW	TXTHIMASK	1’b1

		TXFIFO 阈值中断使能位 1'b0：中断不使能 1'b1：中断使能 当使能中断，并且 TXFIFO 中的数据数等于或者小于 TXTH 时，产生中断	
[5]	RW	TXOVIMASK TXFIFO 溢出中断使能位 1'b0：中断不使能 1'b1：中断使能 注意：当使能中断，TXFIFO 满，CPU 再向 TXFIFO 中写数据时，TXOVIF 标志位将会被置位	1'b1
[4]	RW	TXUDIMASK TXFIFO 下溢中断使能位 1'b0：中断不使能 1'b1：中断使能 注意：当使能 TXFIFO 下溢中断，并且检测到 TXUDIF 为 1 时，将产生下溢中断	1'b1
[3]	RW	RXDONEMASK 接收完成中断使能标志位 1'b0：中断不使能 1'b1：中断使能 当使能接收完成中断，并且接收过程完成时，将产生接收完成中断	1'b1
[2]	RW	RXTHIMASK RXFIFO 阈值中断使能标志位	1'b1

		1’b0：中断不使能 1’b1：中断使能 当使能 RXFIFO 阈值中断，并且 RXFIFO 中的数据个数等于或者多于阈值数时，产生生 RX 中断	
[1]	RW	RXOVIMASK RXFIFO 溢出中断使能位 1’b0：中断不使能 1’b1：中断使能 注意：当使能 RXFIFO 流出中断，并且检测到 TXOVIF 为 1 时，将产生溢出中断	1’b1
[0]	RW	RXUDIMASK RXFIFO 下溢中断使能位 1’b0：中断不使能 1’b1：中断使能 注意：当使能 RXFIFO 下溢中断，并且检测到 TXUDIF 为 1 时，将产生下溢中断	1’b1

16.9.4 中断标志寄存器

表 133 I2S 中断标志寄存器

位	访问	操作说明	复位值
[31:13]	RO	RSV	19’h0
[12]	RO	TXIF I2S 发送中断标志 1’b0：未发生 I2S 中断	1’b0

		1’b1：I2S 有发送中断产生	
[11]	RO	RXIF I2S 接收中断标志 1’b0：未发生 I2S 中断 1’b1：I2S 有接收中断产生	1’b0
[10]	RO	I2SIF I2S 中断标志位 1’b0：未发生 I2S 中断 1’b1：I2S 有中断产生 注意：只要 RX 或者 TX 之一有中断，此位就会置位	1’b0
[9]	RW	LZCIF 左声道零交叉检测标志 此位指示左通道下一个样本数据符号位改变或所有数据位为零。 1’b0：未检测到零交叉 1’b1：检测到零交叉 注意：写 1 来清除中断标志	1’b0
[8]	RW	RZCIF 右声道零交叉检测标志 此位指示右通道下一个样本数据符号位改变或所有数据位为零。 1’b0：未检测到零交叉 1’b1：检测到零交叉 注意：写 1 来清除中断标志	1’b0

[7]	RW	TXDONEIF 发送完成中断标志 1'b0：本次发送未完成 1'b1：本次发送完成 注意：写 1 来清除中断标志	1'b0
[6]	RO	TXTHIF RXFIFO 中断标志 1'b0：TXFIFO 中的字个数大于阈值 1'b1：TXFIFO 中的字个数小于或等于于阈值. 注意：当 TXFIFO 中字的个数(TXCNT)等于或者少于 TXTH 设置的阈值时，这个位会被置 1，直至向 TXFIFO 中的写入数据并且 TXCNT 的值大于 TXTH 的值后，它才会变回 0。	1'b0
[5]	RW	TXOVIF TXFIFO 溢出中断标志 1'b0：TXFIFO 没有发生溢出中断 1'b1：TXFIFO 发生了溢出中断 注意：写 1 来清除中断标志	1'b0
[4]	RW	TXUDIF TXFIFO 下溢中断标志 1'b0：TXFIFO 没有发生下溢中断 1'b1：TXFIFO 发生了下溢中断 注意：写 1 来清除中断标志	1'b0

[3]	RW	RXDONEIF 接收完成中断标志 1'b0：本次接收未完成 1'b1：本次接收完成 注意：写 1 来清除中断标志	1'b0
[2]	RO	RXTHIF RXFIFO 中断标志 1'b0：RXFIFO 中的字个数小于阈值 1'b1：RXFIFO 中的字个数等于或大于阈值. 注意：当 RXFIFO 中字的个数等于或者多于 RXTH 设置的阈值时，这个位会被置 1， 直至 RXFIFO 中的数据被读出并且 RXCNT 的值小于 RXTH 的值后，它才会变回 0。	1'b0
[1]	RW	RXOVIF RXFIFO 溢出中断标志 1'b0：RXFIFO 没有发生溢出中断 1'b1：RXFIFO 发生了溢出中断 注意：写 1 来清除溢出中断	1'b0
[0]	RW	RXUDIF RXFIFO 下溢中断标志 1'b0：RXFIFO 没有发生下溢中断 1'b1：RXFIFO 发生了下溢中断 注意：写 1 来清除下溢中断	1'b0

16.9.5 状态寄存器

表 134 I2S 状态寄存器

位	访问	操作说明	复位值
[31:10]	RO	RSV	22'h0
[9:8]	RO	VALIDBYTE 最后一个字中可用的字节数。 2'b00：接收完成后，RXFIFO 中所有的字节都是可用的 2'b01：接收完成后，RXFIFO 中有 1 个字节是可用的 2'b10：接收完成后，RXFIFO 中有 2 个字节是可用的 2'b11：接收完成后，RXFIFO 中有 3 个字节是可用的	2'h0
[7:4]	RO	TXCNT 记录当前时刻 TXFIFO 中字的个数。 4'b0000：没有数据 4'b0001：有 1 个字 … 4'b1000：有 8 个字	4'h0
[3:0]	RO	RXCNT 记录当前时刻 RXFIFO 中字的个数。 4'b0000：没有数据 4'b0001：有 1 个字 … 4'b1000：有 8 个字	4'h0

16.9.6 数据发送寄存器

表 135 I2S 数据发送寄存器

位	访问	操作说明	复位值
[31:0]	WO	TXFIFO I2S 内置了 8 个字长度的 FIFO 用于存贮待发送的数据。每次向 TXFIFO 中写一个字，TXFIFO 中的字就增加一个。I2S 控制器会自动将先进入 TXFIFO 中的字发送出去。	32'h0

16.9.7 数据接收寄存器

表 136 I2S 数据接收寄存器

位	访问	操作说明	复位值
[31:0]	RO	RXFIFO I2S 内置了 8 个字长度的 FIFO 用于存贮接收到的数据。每次从 RXFIFO 中读取一个字，RXFIFO 中的字就会少一个。	32'h0

17 UART 模块

17.1功能概述

UART 是一种通用串行数据总线，用于异步通信。该总线支持双向通信，可以实现全双工传输和接收。

W800 共 6 组普通 UART 口，通过精细的时钟分频组合可以实现各种波特率的设置，最大可支持 2Mbps 的通信速率。W800 UART 能和硬件 DMA 配合使用，实现数据的高效异步传输。

17.2主要特性

- 符合 APB 总线接口协议，全双工异步通信方式
- 支持中断或轮询工作方式
- 支持 DMA Byte 传输模式，发送接收各 32-byte FIFO
- 波特率可编程，最大支持 2Mbps
- 5-8bit 数据长度，以及 parity 极性可配置
- 1 或 2 个 stop 位可配置
- 支持 RTS/CTS 流控
- 支持 Break 帧发送与接收
- 支持 Overrun，parity error，frame error，rx break frame 中断指示

17.3功能描述

17.3.1 UART 波特率

异步通信因为两边没有同一时钟源作参考，需要通信双方按照协商好的波特率来发送和接收数据。W800 可以通过波特率设置寄存器 BAUD_RATE_CTRL 寄存器来实现精细的波特率控制。

BAUD_RATE_CTRL[15:0]命名为 ubdiv，BAUD_RATE_CTRL[19:16]命名为 ubdiv_frac，需要设置的波特率 baudrate，计算公式如下：

```
ubdiv = apbclk / (16 * baudrate) – 1                    //取整数
ubdiv_frac = (apbclk % (baudrate * 16)) / baudrate   //取整数
```

以 APB 时钟 40MHz，波特率 19200bps 为例：

```
ubdiv = 40000000 / (16 * 19200) – 1 = 129
ubdiv_frac = (40000000 % (19200* 16)) / 19200 = 3
```

根据以上公式计算 APB 时钟 40MHz，波特率 19200bps 的情况下，波特率寄存器应该设置为：BAUD_RATE _CTRL = (3<<16) | 129 = 0x0003_0081。

17.3.2 UART 数据格式

- **数据长度**

W800 的 UART 支持支持 5bit、6bit、7bit、8bit 的数据长度可配置。关于数据长度的定义如下：

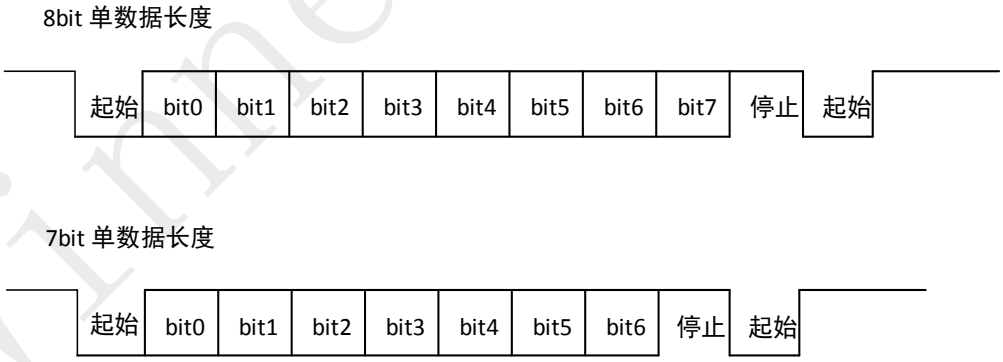


图 28 UART 数据长度

正常的 UART 通信是由 1bit 起始位，1bit 停止位再加中间的数据位，而中间的数据位是可以配置的，W800 支持 5bit、6bit、7bit、8bit 4 种长度的数据位可配置，可根据实际应用来选择数据位长度。

- **停止位**

W800 的 UART 支持 1bit 停止位和 2bit 停止位可配置，可根据实际需要配置，如下：

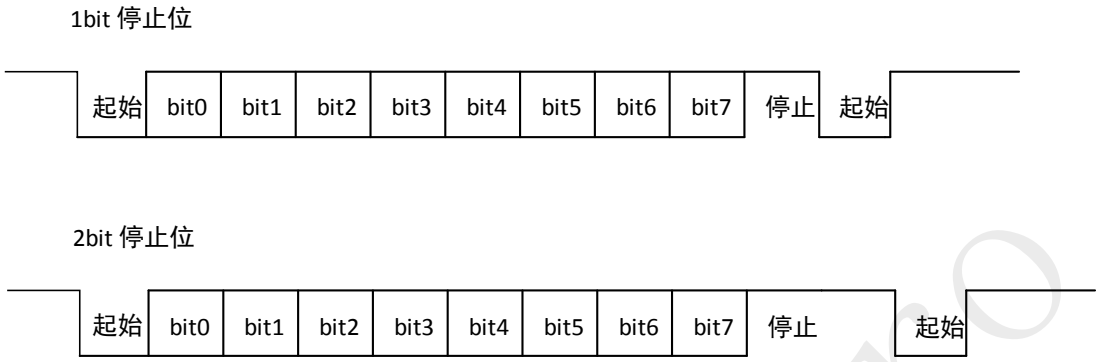


图 29 UART 停止位

- **奇偶检验位**

奇偶校验位的作用是为了校验数据的正确性，W800 可以设置奇校验、偶校验和无校验。

奇校验的计算方法：如果当前数据位 1 的个数是奇数个，奇校验位为 0，如果当前数据为 1 的个数是偶数个，奇校验位为 1。总之保证奇数个 1。

偶校验的计算方法：如果当前数据位 1 的个数是奇数个，偶校验位为 1，如果当前数据为 1 的个数是偶数个，偶校验位为 0。总之保证偶数个 1。

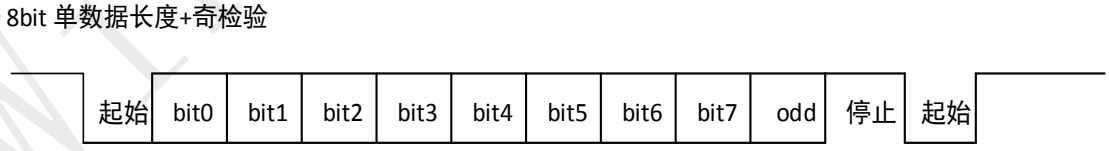
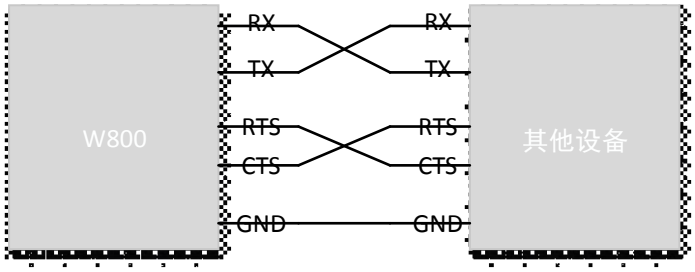


图 30 UART 奇偶校验位

17.3.3 UART 硬件流控

W800 UART 支持 RTS/CTS 方式的硬件流控。流控的主要目的是为了防止 UART fifo 里



的数据因为软件来不及处理而造成丢失，RTS 和 CTS 是对应使用的，如下图：

图 31 UART 硬件流控连接

W800 的硬件流控是通过 AUTO_FLOW_CTRL 寄存器来控制的。当硬件流控 AUTO_FLOW_CTRL[0]为 1 时，W800 会根据 AUTO_FLOW_CTRL[4:2]设定的 rxfifo 中的数据个数来进行流控设置，大于设定个数，RTS 拉高，其他设备则不再给 W800 发送数据，小于设定个数 RTS 拉低，其他设备继续给 W800 发送数据。当 AUTO_FLOW_CTRL[0]为 0 时，软件通过 AUTO_FLOW_CTRL[1]来设定 RTS 的高低。

W800 发送数据时，可以通过中断判断当前 CTS 是否发生变化，并通过 FIFO_STATUS[12]查询 CTS 状态，来决定是否继续给其他设备发送数据。

17.3.4 UART DMA 传输

W800 的 UART 支持 DMA 传输模式，DMA 传输时需要配置 UART 寄存器列表里面的 DMA_CTRL 寄存器来打开 UART 的 DMA 使能。同时需要配置 UART_FIFO_CTRL 配置 txfifo、rxfifo 中剩余多少个字节触发 DMA 搬运。

DMA 的源或目的地址设置为 TX_DATA_WINDOW 或 RX_DATA_WIDNOW，其他 DMA 寄存器的设置参考 DMA 寄存器章节。

注意：UART DMA 传输只能设置为 Byte 模式，不支持 half_word 和 word 传输模式。

17.3.5 UART 中断

UART 支持中断操作模式，包括 fifo 空，fifo 达到设定触发值，CTS 变化，出错等都会产生 UART 中断，可以通过 INT_MASK 寄存器来设置需要的中断。

当 UART 中断产生后，可以通过 INT_SRC 来查询当前的中断状态，触发中断的原因。软件写 1 清 0。

17.4寄存器描述

17.4.1 寄存器列表

表 137 UART 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	数据流控制寄存器	UART_LINE_CTRL	RW	uart 通信的数据格式设置	0X0000_000B
0X0004	自动硬件流控寄存器	AUTO_FLOW_CTRL	RW	uart rts/cts 硬件流控设置	0X0000_0014
0X0008	DMA 设置寄存器	DMA_CTRL	RW	uart dma 传输模式设置	0X0000_0024
0X000C	FIFO 控制寄存器	UART_FIFO_CTRL	RW	设置 uart fifo 触发等级	0X0000_0014
0X0010	波特率控制寄存器	BAUD_RATE_CTRL	RW	设置 uart 通信波特率	0X0003_0081
0X0014	中断屏蔽寄存器	INT_MASK	RW	设置 uart 需要使用的中断	0X0000_01FF
0X0018	中断状态寄存器	INT_SRC	RW	uart 中断状态指示	0X0000_0000

0X001C	FIFO 状态寄存器	FIFO_STATUS	RW	fifo 状态，cts 状态查询	0X0000_1000
0X0020	TX 起始地址寄存器	TX_DATA_WINDOW	WO		0X0000_0000
0X0024	保留				
0X0028	保留				
0X002C	保留				
0X0030	RX 起始地址寄存器	RX_DATA_WINDOW	RO		0X0000_0000
0X0034	保留				
0X0038	保留				
0X003C	保留				

17.4.2 数据流控制寄存器

表 138 UART 数据流控制寄存器

位	访问	操作说明	复位值
[31: 8]		保留	
[7]	RW	uart_rx_enable 接收使能，高有效	1'b0
[6]	RW	uart_tx_enable 发送使能，高有效。	1'b0
[5]	RW	send break enable 发送 break 数据包。Uart 会在该为被置位后发送完一个 break 数据包，发送完成后自动清 0。	1'b0
[4]	RW	parity 极性	1'b0

		1’b0：偶校验 1’b1：奇校验	
[3]	RW	parity en 奇偶检验使能，高有效	1’b1
[2]	RW	stop bit 个数 1’b0：1 个停止位 1’b1：2 个停止位	1’b0
[1 : 0]	RW	uart 比特长度。 2’h0：5bit 2’h1：6bit 2’h2：7bit 2’h3：8bit	2’h3

17.4.3 自动硬件流控寄存器

表 139 UART 自动硬件流控寄存器

位	访问	操作说明	复位值
[31: 5]		保留	
[4 : 2]	RW	RTS trigger level 在 afc_enable 有效时，决定何时需要将 RTS 置无效。 3’h0：rxfifo 有 4 个以上字节 3’h1：rxfifo 有 8 个以上字节	3’h5

		3'h2：rxfifo 有 12 个以上字节 3'h3：rxfifo 有 16 个以上字节 3'h4：rxfifo 有 20 个以上字节 3'h5：rxfifo 有 24 个以上字节 3'h6：rxfifo 有 28 个以上字节 3'h7：rxfifo 有 31 个以上字节	
[1]	RW	RTS set 当 AFC_enable 无效时，软件可以通过设置此位来完成接收流量控制。当 AFC_enable 有效时，此位不关心。	1'b0
[0]	RW	afc enable 1'b1：有效，接收条件 rts 使用 rts_trigger_level 控制产生。	1'b0

17.4.4 DMA 设置寄存器

表 140 UART DMA 设置寄存器

位	访问	操作说明	复位值
[31: 8]		保留	
[7 : 3]	RW	rxfifo timeout num rxfifo 中存在数据小于 rxfifo_trigger_level 情况下，如果 N 个包的时间内，没有接收到新的数据时，产生 rxfifo timeout 中断。 该计时功能使能后，无论是第一次计时还是上一次计时完成，都只在接收到至少 1 个包后才开始计时	5'h04
[2]	RW	rxfifo timeout en	1'b1

		rxfifo 超时使能	
[1]	RW	rx dma enable 接收 DMA 使能，高有效。 0：表示接收过程使用中断。	1'b0
[0]	RW	tx dma enable 发送 DMA 使能，高有效。 0：表示发送过程使用中断。	1'b0

17.4.5 FIFO 控制寄存器

表 141 UART FIFO 控制寄存器

位	访问	操作说明	复位值
[31: 6]		保留	
[5 : 4]	RW	rxfifo trigger level 当 rxfifo 中数据字节数大于等于该值时，触发中断，或者触发 rxdma req。 2'h0：1byte 2'h1：4byte 2'h2：8byte 2'h3：16byte	2'h1
[3 : 2]	RW	txfifo trigger level 当 txfifo 中数据字节数小于等于该值时，触发中断，或者触发 txdma req。 2'h0：empty 2'h1：4byte	2'h1

		2'h2：8byte 2'h3：16byte	
[1]	RW	rxfifo reset 复位 rxfifo，将 rxfifo 状态清空	1'b0
[0]	RW	txfifo reset 复位 txfifo，将 txfifo 状态清空	1'b0

17.4.6 波特率控制寄存器

表 142 UART 波特率控制寄存器

位	访问	操作说明	复位值
[31:20]		保留	
[19:16]	RW	ubdiv_frac 系统时钟除以 16 倍波特率时钟商的小数部分指示。具体值为 frac×16。 （参考章节 2.3.2，波特率计算方法）	4'h3
[15: 0]	RW	ubdiv 系统时钟除以 16 倍波特率时钟商的整数部分减 1。 默认系统时钟为 40MHz，波特率为 19200。 （参考章节 2.3.2，波特率计算方法）	16'h81

17.4.7 中断屏蔽寄存器

表 143 UART 中断屏蔽寄存器

位	访问	操作说明	复位值

[31: 9]		保留	
[8]	RW	overrun error int mask， rxfifo 溢出中断屏蔽位，高有效。	1’b1
[7]	RW	parity error int mask，奇偶检验中断屏蔽位，高有效。	1’b1
[6]	RW	frame error int mask，数据帧出错中断屏蔽位，高有效。	1’b1
[5]	RW	break detect int mask，break 信号检测中断屏蔽位，高有效。	1’b1
[4]	RW	cts changed indicate mask，CTS 信号变化中断屏蔽位，高有效。	1’b1
[3]	RW	rxfifo data timeout int mask，rxfifo 接收数据超时中断屏蔽位，高有效。	1’b1
[2]	RW	rxfifo trigger level int mask，rxfifo 达到触发值中断屏蔽位，高有效。	1’b1
[1]	RW	txfifo trigger level int mask，txfifo 达到触发值中断屏蔽位，高有效。	1’b1
[0]	RW	txfifo empty int mask，txfifo 为空中断屏蔽位，高有效。	1’b1

17.4.8 中断状态寄存器

表 144 UART 中断状态寄存器

位	访问	操作说明	复位值
[31: 9]		保留	
[8]	RW	overrun error rxfifo 出现溢出。 软件主动写 1 清 0。	1’b0
[7]	RW	parity error 接收到的包校验位错误。 DMA 情况下，此中断仍会产生。但 DMA 操作不关心此中断。 软件主动写 1 清 0。	1’b0

[6]	RW	frame error 接收到的包停止位错误。 DMA 情况下，此中断仍会产生。但 DMA 操作不关心此中断。 软件主动写 1 清 0。	1'b0
[5]	RW	break detect 接收到 break 包。 DMA 情况下，此中断仍会产生。但 DMA 操作不关心此中断。 软件主动写 1 清 0。	1'b0
[4]	RW	cts changed cts 信号变化则产生此中断。 软件主动写 1 清 0。	1'b0
[3]	RW	rxfifo data timeout rxfifo 中数据长度小于 rxfifo trigger level 但 N 个数据周期没有接收到任何数据， 则产生中断。 软件主动写 1 清 0。	1'b0
[2]	RW	rxfifo trigger level interrupt 当 rxfifo 中数据个数由小于 rxfifo trigger level 中指定的数变成大于或等于该数 时，产生此中断。 此时应该根据 rxfifo count 确定当前数据帧大小。 软件主动写 1 清 0。	1'b0
[1]	RW	txfifo trigger level interrupt 当 txfifo 中数据个数由大于 txfifo trigger level 中指定的数变成小于或等于该数	1'b0

		时，产生中断。 软件主动写 1 清 0。	
[0]	RW	tx fifo empty interrupt 当发送完成当前包，并且 txfifo 为空时，产生此中断。 软件主动写 1 清 0。	1'b0

17.4.9 FIFO 状态寄存器

表 145 UART FIFO 状态寄存器

位	访问	操作说明	复位值
[31:13]		保留	
[12]	RW	cts status 当前 cts 的状态	1'b0
[11: 6]	RW	rxfifo count rxfifo 中数据个数	6'h0
[5 : 0]	RW	txfifo count txfifo 中数据个数	6'h0

17.4.10TX 起始地址寄存器

表 146 UART TX 起始地址寄存器

位	访问	操作说明	复位值
[31: 0]	WO	tx data window 发送数据起始地址。	32'h0

		注意：uart 发送与接收数据只支持字节操作，当采用 burst 传输时，有可能使用字节地址递增的方式，设计中最多支持 16-burst 的操作，即 16byte。因此从发送/接收起始地址后共 16byte（4 个字）都保留为发送/接收数据窗口。	

17.4.11RX 起始地址寄存器

表 147 UART RX 起始地址寄存器

位	访问	操作说明	复位值
[31: 0]	RO	rx data window 接收数据起始地址。 注意：uart 发送与接收数据只支持字节操作，当采用 burst 传输时，有可能使用字节地址递增的方式，设计中最多支持 16-burst 的操作，即 16byte。因此从发送/接收起始地址后共 16byte（4 个字）都保留为发送/接收数据窗口。	32'h0

18 UART&7816 模块

18.1 功能概述

UART&7816 模块兼容 UART 功能，同时兼容 7816 接口功能。

W800 支持 1 组 UART&7816 复用接口(uart2)，作为 UART 使用时，可以通过精细的时钟分频组合可以实现各种波特率的设置，最大可支持 2Mbps 的通信速率。

W800 UART&7816 接口能和硬件 DMA 配合使用，实现数据的高效传输。

18.2 主要特性

- 符合 APB 总线接口协议，支持 UART 异步全双工和 7816 异步半双工通信方式；
- 支持中断或轮询工作方式；
- 支持 DMA Byte 传输模式，发送接收各 32-byte FIFO；
- 串口功能：
  - 波特率可编程，最大支持 2Mbps
  - 5-8bit 数据长度，以及 parity 极性可配置
  - 1 或 2 个停止位可配置
  - 支持 RTS/CTS 流控
  - 支持 Break 帧发送与接收
  - 支持 Overrun，parity error，frame error，rx break frame 中断指示
- 7816 接口功能：
  - 兼容 ISO-7816-3 T=0.T=1 模式

- 兼容 EVM2000 协议
- 可配置 guard time（11 ETU-267 ETU）
- 正向/反向约定，可软件配置
- 支持发送/接收奇偶校验及重传功能
- 支持 0.5 和 1.5 个 stop 位可配置

18.3 UART 功能描述

参考 16 章 UART 功能模块描述

18.4 7816 功能描述

18.4.1 7816 简介

ISO7816 是国际标准的智能卡协议，协议规定了智能卡物理特性、尺寸和接口、电信号和传输协议、命令、安全等多方面信息。

W800 主要实现了 ISO 7816 -3 电信号和传输协议这部分，支持 T0 和 T1 卡。通过 W800 的 7816 接口，用户可以无法关心时钟和数据的信号通信逻辑，可以直接与智能卡进行数据和命令的交互。关于智能卡的数据和命令交互方式，用户需要自己参考 ISO7816-4 协议去实现。

18.4.2 7816 接口

W800 主要集成了智能卡的时钟和数据两个接口，来实现数据和命令的通信电信号逻辑。实际智能卡应用中，还有 RST、VCC、GND 这三个信号，RST 可以通过普通 GPIO 来控制，主要是智能卡上电复位时使用。VCC 可以直接连接 3.3V 电源，或者通过 GPIO 配合其他电路来控制智能卡 VCC 的通断。

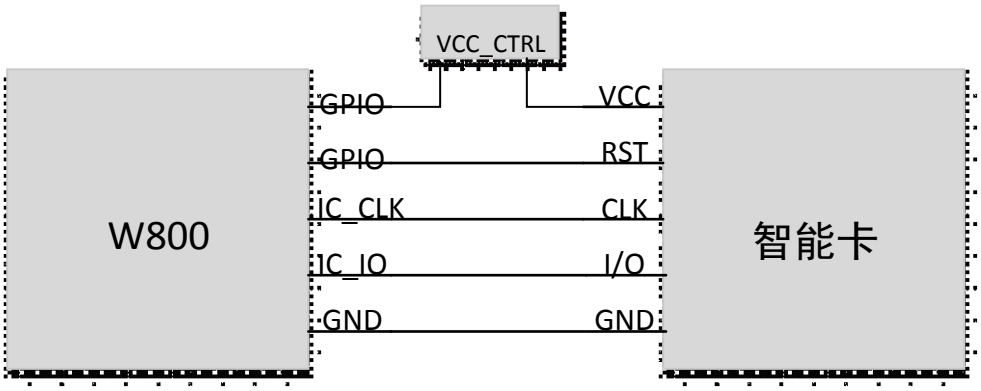


图 32 7816 连接示意图

18.4.3 7816 配置

作为 7816 接口使用时需要进行相关配置：

- 设置接口工作模式，UART_LIN_CTRL[24]设置为 1，选择当前接口为 7816 模式；
- 设置 7816 MSB 或 LSB 传输模式，UART_LIN_CTRL[3]设置为 1，通过 UART_LIN_CTRL[3]来选择 7816 接口是 MSB 模式(bit7 先传输)，还是 LSB 模式(bit0 先传输)；
- 设置停止位，UART_LIN_CTRL[2]可选择智能卡 0.5 或者 1.5 个停止位；
- 选择卡片类型，UART_LIN_CTRL[8]可以选择 T0 卡或者 T1 卡；
- 配置智能卡通信超时时间，通过 WAIT_TIME 来设置超时时间，接收数据时超时未收到数据则产生超时中断。

18.4.4 7816 时钟配置

智能卡时钟是指通过 CLK 引脚提供给智能卡的时钟，通过 BAUD_RATE_CTRL[21:16]来设置，计算方法如下：

$$\text{clk\_div} = \frac{fclk\_apb}{2 \times fsc\_clk} - 1$$

fsc_clk：需要给智能卡提供的 CLK；

fclk_apb：系统 APB 总线时钟；

clk_div：需要设置给 BAUD_RATE_CTRL[21:16]的时钟分频因子

因为 clk_div 只能取整数，为了减少误差，我们最好采取四舍五入的计算方法，C 语言计算取整会丢掉小数部分，C 语言采取四舍五入的转换方法如下：

clk_div = (fclk_apb + fsc_clk)/(2 * fsc_clk) - 1;

18.4.5 7816 速率设置

智能卡中有一个时间单位 ETU，智能卡按此时间单位来传输数据和命令。ETU 的设置是通过 BAUD_RATE_CTRL[15: 0]来设置的，计算方法如下：

$$1\text{etu} = \frac{F}{D} \times \frac{1}{f}$$

f：即我们智能卡的 CLK；

F 和 D 均是有智能卡给出的参数。

其实我们需要设置的 BAUD_RATE_CTRL[15: 0]的 ubdiv 就是 F/D，上面的公式只是为了计算 ETU 供大家参考。我们实际设置的时候，只需要设置 ubdiv = F/D 即可。F 和 D 可由下表进行查询。

Table 7 — *Fi* and *f* (max.)

Bits 8 to 5	0000	0001	0010	0011	0100	0101	0110	0111
Fi	372	372	558	744	1116	1488	1860	RFU
f **(max.) MHz**	4	5	6	8	12	16	20	—

Bits 8 to 5	1000	1001	1010	1011	1100	1101	1110	1111
Fi	RFU	512	768	1024	1536	2048	RFU	RFU
f **(max.) MHz**	—	5	7,5	10	15	20	—	—

— According to Table 8, bits 4 to 1 encode *Di*.

Table 8 — *Di*

Bits 4 to 1	0000	0001	0010	0011	0100	0101	0110	0111
Di	RFU	1	2	4	8	16	32	64

Bits 4 to 1	1000	1001	1010	1011	1100	1101	1110	1111
Di	12	20	RFU	RFU	RFU	RFU	RFU	RFU

表 148 7816 速率设置

（参考协议文件 ISO_IEC_FDIS_7816-3_(E).PDF）

18.4.6 7816 上电复位

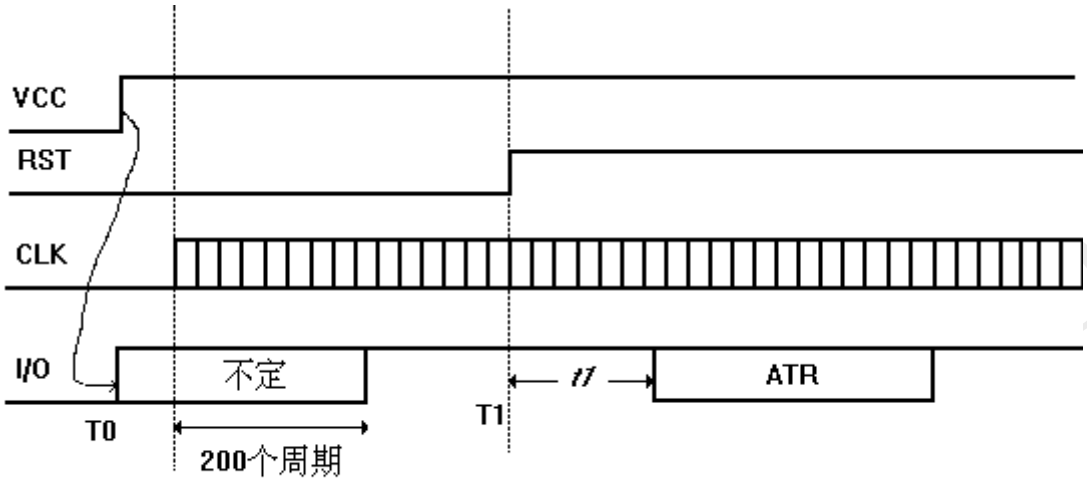


图 33 7816 上电复位时序

上图为智能卡上电复位的时序图。CLK 和 IO 初始状态为低，需要我们配置成 GPIO 模式并拉低。VCC 拉高后 CLK 和 IO 配置成 7816 模式后由 7816 控制即可。最后我们需要把 RST 引脚手动拉高，完成复位过程。配置步骤如下：

- I/O、CLK、RST 配置为普通 GPIO 模式并保持低电平；
- 设置 7816 为 T=0 模式；
- 通过 GPIO 控制 VCC 上电；
- 配置 I/O、CLK 为 7816 模式，由 7816 驱动时钟和数据；
- 配置 7816 时钟频率并允许时钟输出；
- 置位 RST 管脚，等待接收 ATR 数据，若 40000 个时钟内没有收到 ATR 数据，则执行失活流程，卡片失活。

18.4.7 7816 热复位

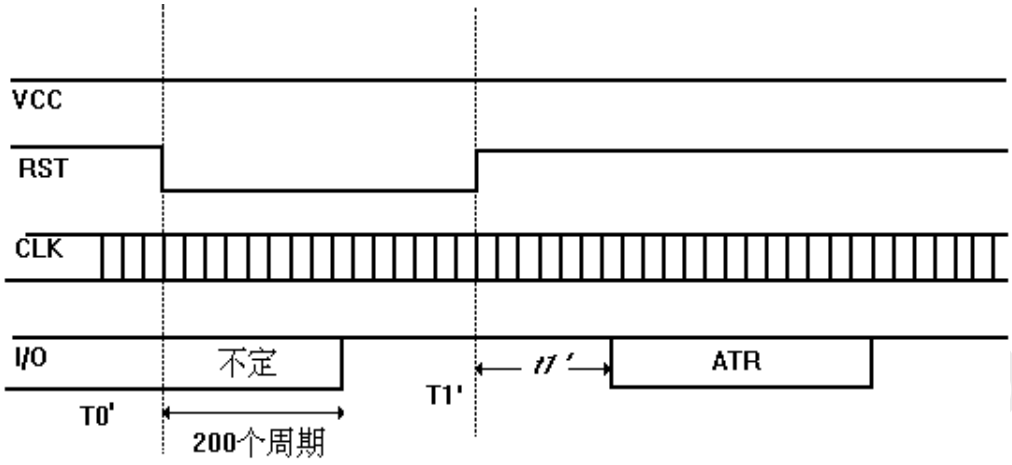


图 34 7816 热复位

如上图所示，热复位的过程很简单，正常工作模式，将 RST 引脚拉低 400 个周期即可。配置步骤如下：

- 保持 VCC 上电状态；
- 拉低 RST 引脚至少 400 个时钟周期；
- 拉高 RST 引脚，等待接收 ATR 数据，若 40000 个时钟内没有收到 ATR 数据，则执行失活流程，卡片失活。

18.4.8 7816 失活过程

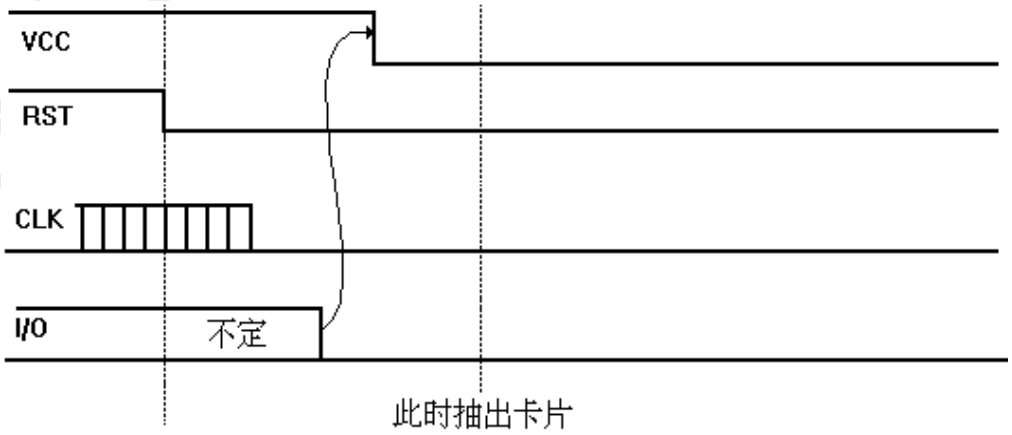


图 35 7816 失活过程

如上图所示，RST 拉低之后需要将 CLK 和 IO 配置成普通 IO 模式并拉低，最后关闭 VCC 电源,操作步骤如下：

- 保持 VCC 上电状态；
- 拉低 RST 引脚；
- 配置 CLK 和 IO 为 GPIO 模式，并拉低；
- 通过 GPIO 控制 VCC 掉电；

18.4.9 7816 数据传输

7816 的数据传输的时序已经有 W800 硬件完成，无需用户操作，用户如果想要了解此部分具体内容，请参考 ISO7816-3 协议中的规定。

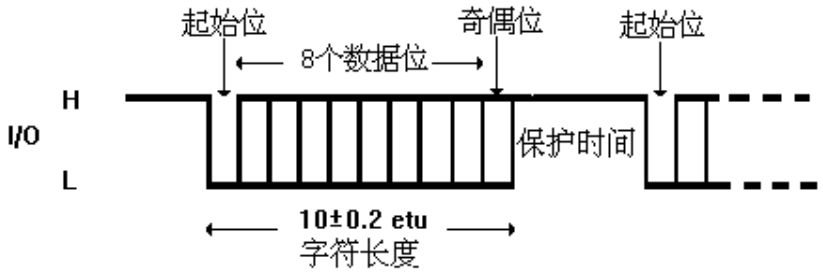


图 36 7816 数据传输

18.4.10UART&7816 DMA 传输

W800 的 UART&7816 支持 DMA 传输模式，DMA 传输时需要配置 UART&7816 寄存器列表里面的 DMA_CTRL 寄存器来打开 UART 的 DMA 使能。同时需要配置 UART_FIFO_CTRL 配置 txfifo、rxfifo 中剩余多少个字节触发 DMA 搬运。

DMA 的源或目的地址设置为 TX_DATA_WINDOW 或 RX_DATA_WIDNOW，其他 DMA 寄存器的设置参

考 DMA 寄存器章节。

注意：UART&7816 DMA 传输只能设置为 Byte 模式，不支持 half_word 和 word 传输模式。

18.4.11UART&7816 中断

UART&7816 支持中断操作模式，包括 fifo 空，fifo 达到设定触发值，CTS 变化，出错等都会产生 UART&7816 中断，可以通过 INT_MASK 寄存器来设置需要的中断。

当 UART&7816 中断产生后，可以通过 INT_SRC 来查询当前的中断状态，触发中断的原因。软件写 1 清 0。

18.5 寄存器描述

18.5.1 寄存器列表

表 149 UART&7816 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	数据流控制寄存器	UART_LINE_CTRL	RW	uart&7816 通信的数据相关设置	0X0033_520B
0X0004	自动硬件流控寄存器	AUTO_FLOW_CTRL	RW	uart rts/cts 硬件流控设置	0X0000_0014
0X0008	DMA 设置寄存器	DMA_CTRL	RW	uart&7816 dma 传输模式设置	0X0000_0024
0X000C	FIFO 控制寄存器	UART_FIFO_CTRL	RW	设置 uart&7816 fifo 触发等级	0X0000_0014
0X0010	波特率控制寄存器	BAUD_RATE_CTRL	RW	设置 uart 波特率、7816 时钟	0X0003_0082
0X0014	中断屏蔽寄存器	INT_MASK	RW	设置 uart&7816 需要使用的中断	0X0000_03FF

0X0018	中断状态寄存器	INT_SRC	RW	uart&7816 中断状态指示	0X0000_0000
0X001C	FIFO 状态寄存器	FIFO_STATUS	RW	fifo 状态，cts 状态查询	0X0000_0000
0X0020	TX 起始地址寄存器	TX_DATA_WINDOW	WO		0X0000_0000
0X0024	保留				
0X0028	保留				
0X002C	保留				
0X0030	RX 起始地址寄存器	RX_DATA_WINDOW	RO		0X0000_0000
0X0034	保留				
0X0038	保留				
0X003C	保留				
0X0040	7816 保护时间寄存器	GUARD_TIME	RW	7816 数据间保护时间	0X0000_0000
0X0044	7816 超时时间寄存器	WAIT_TIME	RW	7816 接收数据超时时间	0X0007_8000

18.5.2 数据流控制寄存器

表 150 UART&7816 数据流控制寄存器

位	访问	操作说明	复位值
[31:25]		保留	
[24]	RW	sc_mode 1’b0：uart 模式 1’b1：7816 模式	1’b0
[23]	RW	7816 卡 T0 模式 rx_retrans_en 1’b0：Rx 自动重传无效	1’b0

		1’b1：Rx 自动重传使能	
[22:20]	RW	7816 卡 T0 模式 rx_retrans_cnt	3’h3
[19]	RW	7816 卡 T0 模式 tx_retrans_en 1’b0：Tx 自动重传无效 1’b1：Tx 自动重传使能	1’b0
[18:16]	RW	7816 卡 T0 模式 tx_retrans_cnt tx 自动重传次数	3’h3
[15:11]	RW	7816 卡的最小 MIN_BGT（Min Block Guard Time ） Min Block Guard Time 计算： 10+stop 位(默认 2 位)+配置值 MIN_BGT Note: T=0：在发送和接收的两个相反方向连续字符的起始位下降沿之间的最小时间间隔不能小于 16 个 ETU。必须能够正确解释接收到的其起始位下降沿和最后发送的字节起始位下降沿间隔为 15 个 ETU 的字符。 T=1：在发送和接收的两个相反方向连续字符的起始位下降沿之间的最小时间间隔（块保护时间，BGT）必须为 22 个 ETU。必须能够正确解释接收到的其起始位下降沿和最后发送的字节起始位下降沿间隔为 21 个 ETU 以内接收到的字符。	5’ha
[10]	RW	7816 卡时钟控制配置 1’b0：在配置为卡模式时产生卡时钟输出，否则卡时钟输出无效 1’b1：时钟停止	1’b0
[9]	RW	7816 卡的 parity 错误时是否接收数据 1’b0：不接收 1’b1：接收	1’b1

[8]	RW	7816 卡的 T0/T1 模式配置， 1'b0：T0 模式 1'b1：T1 模式	1'b0
[7]	RW	uart_rx_enable uart/7816 模式下，接收使能，高有效	1'b0
[6]	RW	uart_tx_enable uart/7816 模式下，发送使能，高有效。	1'b0
[5]	RW	send break enable 发送 break 数据包。Uart 会在该为被置位后发送完一个 break 数据包，发送完成后自动清 0。	1'b0
[4]	RW	parity 极性（UART 模式） 1'b0：偶校验 1'b1：奇校验 正反向(7816 模式) 1'b0：LSB（b0 bit）先传输 1'b1：MSB（b7 bit）先传输	1'b0
[3]	RW	parity 使能，高有效(UART 模式)	1'b1
[2]	RW	stop bit 个数（UART 模式） 1'b0：1 个停止位 1'b1：2 个停止位 stop bit 个数（7816 模式） 1'b0：0.5 个停止位	1'b0

		1'b1：1.5 个停止位	
[1 : 0]	RW	uart 比特长度（UART 模式） 2'h0：5bit 2'h1：6bit 2'h2：7bit 2'h3：8bit	2'h3

18.5.3 自动硬件流控寄存器

表 151 UART&7816 自动硬件流控寄存器

位	访问	操作说明	复位值
[31: 5]		保留	
[4 : 2]	RW	RTS trigger level（UART 模式） 在 afc_enable 有效时，决定何时需要将 RTS 置无效。 3'h0：rxfifo 有 4 个以上字节 3'h1：rxfifo 有 8 个以上字节 3'h2：rxfifo 有 12 个以上字节 3'h3：rxfifo 有 16 个以上字节 3'h4：rxfifo 有 20 个以上字节 3'h5：rxfifo 有 24 个以上字节 3'h6：rxfifo 有 28 个以上字节 3'h7：rxfifo 有 31 个以上字节	3'h5

[1]	RW	RTS set（UART 模式） 当 AFC_enable 无效时，软件可以通过设置此位来完成接收流量控制。当 AFC_enable 有效时，此位不关心。	1'b0
[0]	RW	afc enable（UART 模式） 接收条件 rts 使用 rts_trigger_level 控制产生，高有效。	1'b0

18.5.4 DMA 设置寄存器

表 152 UART&7816 DMA 设置寄存器

位	访问	操作说明	复位值
[31: 8]		保留	
[7 : 3]	RW	rxfifo timeout num（UART 模式） rxfifo 中存在数据小于 rxfifo_trigger_level 情况下，如果 N 个包的时间内，没有 接收到新的数据时，产生 rxfifo timeout 中断。 该计时功能使能后，无论是第一次计时还是上一次计时完成，都只在接收到至少 1 个包后才开始计时	5'h4
[2]	RW	rxfifo timeout en（UART&7816 模式） rxfifo 超时使能，高有效	1'b1
[1]	RW	rx dma enable（UART&7816 模式） 接收 DMA 使能，高有效。 0 表示接收过程使用中断。	1'b0
[0]	RW	tx dma enable（UART&7816 模式） 发送 DMA 使能，高有效。	1'b0

		0 表示发送过程使用中断。	

18.5.5 FIFO 控制寄存器

表 153 UART&7816 FIFO 控制寄存器

位	访问	操作说明	复位值
[31: 6]		保留	
[5 : 4]	RW	rxfifo trigger level(UART&7816 模式) 当 rxfifo 中数据字节数大于等于该值时，触发中断，或者触发 rxdma req。 2'h0：1byte 2'h1：4byte 2'h2：8byte 2'h3：16byte	2'h1
[3 : 2]	RW	txfifo trigger level(UART&7816 模式) 当 txfifo 中数据字节数小于等于该值时，触发中断，或者触发 txdma req。 2'h0：empty 2'h1：4byte 2'h2：8byte 2'h3：16byte	2'h1
[1]	RW	rxfifo reset(UART&7816 模式) 复位 rxfifo， 将 rxfifo 状态清空	1'b0
[0]	RW	txfifo reset(UART&7816 模式) 复位 txfifo，将 txfifo 状态清空	1'b0

18.5.6 波特率控制寄存器

表 154 UART&7816 波特率控制寄存器

位	访问	操作说明	复位值
[31:20]		保留	
[19:16]	RW	ubdiv_frac UART 模式： 系统时钟除以 16 倍波特率时钟商的小数部分指示。具体值为 frac×16。 （参考章节波特率计算方法） 7816 模式： ubdiv_frac = (fclk_apb + fsc_clk)/(2 * fsc_clk) - 1; （参考 7816 时钟计算方法）	4'h3
[15: 0]	RW	ubdiv UART 模式： 系统时钟除以 16 倍波特率时钟商的整数部分减 1。 默认系统时钟为 40MHz，波特率为 19200。 （参考波特率计算方法） 7816 模式： ubdiv=Fi/Di (Fi、Di 为智能卡反馈的参数,edu 频率：f_etuclk = fsc_clk/(ubdiv+1)) （参考章节 7816 速率计算方法）	16'h82

18.5.7 中断屏蔽寄存器

表 155 UART&7816 中断屏蔽寄存器

位	访问	操作说明	复位值
[31:10]		保留	
[9]	RW	7816 卡发送时收到 error signal 错误信号。(7816 模式)	1'b1
[8]	RW	overrun error int mask， rxfifo 溢出中断屏蔽位，高有效。(UART&7816 模式)	1'b1
[7]	RW	parity error int mask，奇偶检验中断屏蔽位，高有效。(UART&7816 模式)	1'b1
[6]	RW	frame error int mask，数据帧出错中断屏蔽位，高有效。(UART 模式)	1'b1
[5]	RW	break detect int mask，break 信号检测中断屏蔽位，高有效。(UART 模式)	1'b1
[4]	RW	cts changed indicate mask，CTS 信号变化中断屏蔽位，高有效。(UART 模式)	1'b1
[3]	RW	rxfifo data timeout int mask，rxfifo 接收数据超时中断屏蔽位，高有效。(UART&7816 模式)	1'b1
[2]	RW	rxfifo trigger level int mask，rxfifo 达到触发值中断屏蔽位，高有效。(UART&7816 模式)	1'b1
[1]	RW	txfifo trigger level int mask，txfifo 达到触发值中断屏蔽位，高有效。(UART&7816 模式)	1'b1
[0]	RW	txfifo empty int mask，txfifo 为空中断屏蔽位，高有效。(UART&7816 模式)	1'b1

18.5.8 中断状态寄存器

表 156 UART&7816 中断状态寄存器

位	访问	操作说明	复位值
[31: 9]		保留	

[9]	RW	7816 卡发送时收到 error signal 错误信号。(7816 模式)	
[8]	RW	overrun error（UART&7816 模式） rxfifo 出现溢出。 软件主动写 1 清 0。	1'b0
[7]	RW	parity error（UART&7816 模式） 接收到的包校验位错误。 DMA 情况下，此中断仍会产生。但 DMA 操作不关心此中断。 软件主动写 1 清 0。	1'b0
[6]	RW	frame error（UART 模式） 接收到的包停止位错误。 DMA 情况下，此中断仍会产生。但 DMA 操作不关心此中断。 软件主动写 1 清 0。	1'b0
[5]	RW	break detect（UART 模式） 接收到 break 包。 DMA 情况下，此中断仍会产生。但 DMA 操作不关心此中断。 软件主动写 1 清 0。	1'b0
[4]	RW	cts changed（UART 模式） cts 信号变化则产生此中断。 软件主动写 1 清 0。	1'b0
[3]	RW	rxfifo data timeout（UART&7816 模式） rxfifo 中数据长度小于 rxfifo trigger level 但 N 个数据周期没有接收到任何数据， 则产生中断。	1'b0

		软件主动写 1 清 0。	
[2]	RW	rxfifo trigger level interrupt（UART&7816 模式） 当 rxfifo 中数据个数由小于 rxfifo trigger level 中指定的数变成大于或等于该数时，产生此中断。 此时应该根据 rxfifo count 确定当前数据帧大小。 软件主动写 1 清 0。	1'b0
[1]	RW	txfifo trigger level interrupt（UART&7816 模式） 当 txfifo 中数据个数由大于 txfifo trigger level 中指定的数变成小于或等于该数时，产生中断。 软件主动写 1 清 0。	1'b0
[0]	RW	tx fifo empty interrupt（UART&7816 模式） 当发送完成当前包，并且 txfifo 为空时，产生此中断。 软件主动写 1 清 0。	1'b0

18.5.9 FIFO 状态寄存器

表 157 UART&7816 FIFO 状态寄存器

位	访问	操作说明	复位值
[31:13]		保留	
[12]	RW	cts status（UART 模式） 当前 cts 的状态	1'b0

[11: 6]	RW	rxfifo count（UART&7816 模式） rxfifo 中数据个数	6'h0
[5 : 0]	RW	txfifo count（UART&7816 模式） txfifo 中数据个数	6'h0

18.5.10 TX 起始地址寄存器

表 158 UART&7816 TX 起始地址寄存器

位	访问	操作说明	复位值
[31: 0]	WO	tx data window（UART&7816 模式） 发送数据起始地址。 注意：uart 发送与接收数据只支持字节操作，当采用 burst 传输时，有可能使用字节地址递增的方式，设计中最多支持 16-burst 的操作，即 16byte。因此从发送/接收起始地址后共 16byte（4 个字）都保留为发送/接收数据窗口。	32'h0

18.5.11 RX 起始地址寄存器

表 159 UART&7816 RX 起始地址寄存器

位	访问	操作说明	复位值
[31: 0]	RO	rx data window（UART&7816 模式） 接收数据起始地址。 注意：uart 发送与接收数据只支持字节操作，当采用 burst 传输时，有可能使用字节地址递增的方式，设计中最多支持 16-burst 的操作，即 16byte。因此从发送/接收起始地址后共 16byte（4 个字）都保留为发送/接收数据窗口。	32'h0

18.5.12 7816 保护时间寄存器

表 160 7816 保护时间寄存器

位	访问	操作说明	复位值
[31: 8]		保留	
[7 : 0]	RW	ex_gt_num 7816 模式下，guard time 计算：10+stop 位+配置值 ex_gt_num	8'h0

18.5.13 7816 超时时间寄存器

表 161 7816 超时时间寄存器

位	访问	操作说明	复位值
[31:24]		保留	
[23: 0]	RW	wait time 计数器（以 ETU 为单位） 7816 模式下： CWT 和 BWT 时间，配置为最大默认值。 （在 T1 模式下：BWT =（11 etu+ 2BWI*960*Fd/fsc））	24'h78000

19 Timer 模块

19.1 功能概述

定时器包含一个 32-bit 自动加载的计数器，该计数器由系统时钟经过分频后驱动。W800 有 6 路完全独立定时器。实现了精确的定时时间以及中断功能，可用于延时或者周期性事件处理。

19.2 主要特性

- 6 路完全独立的定时器
- 32-bit 自动加载计数器
- 定时单位可配置为 ms、us
- 可实现单次定时或者重复定时功能
- 定时中断功能
- 可随时查询计时器计数值；

19.3 功能描述

定时器模块由 6 路完全独立的定时器组成，互不影响，6 路可以同时工作。

系统时钟经过分频系数分频后得到 us 标准时钟，用于计数器的输入时钟。定时单位可配置为 us、ms 两种级别。

定时值是一个 32-bit 可配置的寄存器，可满足不同定时时长的需求。每一个定时器对应一个中断，当定时时间满足后，如果使能了中断功能，则会产生一个中断请求，可以用于处理周期性事件。

19.3.1 定时功能

定时功能是指依据用户设定时间，当时间到产生硬件中断，通知用户实现特定的功能。定时触发支持单次和周期两种，一种可用于处理单次事件，一种可用于处理周期性事件。

用户依据系统时钟的分频系数获得 APB 总线时钟频率，设定定时器的基准微秒计数配置寄存器（TMR_CONFIG），设置定时值，配置定时单位，工作模式，使能中断，然后，启动定时功能。当定时时间到，程序进入定时器中断处理函数，清除中断。

19.3.2 延时功能

延时功能是指用户可以依据定时器的倒计时功能，让程序处于等待状态，直至计时完成，程序才继续运行。

19.4 寄存器描述

19.4.1 寄存器列表

表 162 Timer 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	标准 us 配置寄存器	TMR_CONFIG	RW	标准 us 定时分频值，由总线时钟分频得到标准 us 定时，该值等于 APB 总线频率（MHz）减一	0X0000_0027
0X0004	定时器控制寄存器	TMR_CSR	RW	定时器控制寄存器	0X0631_8C63
0X0008	定时器 1 定时值配置寄存器	TMR1_PRD	RW	Timer1 定时值配置寄存器	0X0000_0000
0X000C	定时器 2 定时值配置寄存器	TMR2_PRD	RW	Timer2 定时值配置寄存器	0X0000_0000
0X0010	定时器 3 定时值配置寄存器	TMR3_PRD	RW	Timer3 定时值配置寄存器	0X0000_0000

0X0014	定时器 4 定时值配置寄存器	TMR4_PRD	RW	Timer4 定时值配置寄存器	0X0000_0000
0X0018	定时器 5 定时值配置寄存器	TMR5_PRD	RW	Timer5 定时值配置寄存器	0X0000_0000
0X001C	定时器 6 定时值配置寄存器	TMR6_PRD	RW	Timer6 定时值配置寄存器	0X0000_0000
0X0020	定时器 1 当前计数值	TMR1_CNT	RO	Timer1 当前计数值	0X0000_0000
0X0024	定时器 2 当前计数值	TMR1_CNT	RO	Timer2 当前计数值	0X0000_0000
0X0028	定时器 3 当前计数值	TMR1_CNT	RO	Timer3 当前计数值	0X0000_0000
0X002C	定时器 4 当前计数值	TMR1_CNT	RO	Timer4 当前计数值	0X0000_0000
0X0030	定时器 5 当前计数值	TMR1_CNT	RO	Timer5 当前计数值	0X0000_0000
0X0034	定时器 6 当前计数值	TMR1_CNT	RO	Timer6 当前计数值	0X0000_0000

19.4.2 标准 us 配置寄存器

表 163 Timer 标准 us 配置寄存器

位	访问	操作说明	复位值
[31: 7]		保留	
[6 : 0]	RW	时钟分频配置 prescale。 例如： apb_clk=40MHz prescale = 40 - 1 = 8'd39	7'h27

19.4.3 定时器控制寄存器

表 164 Timer 定时器控制寄存器

位	访问	操作说明	复位值

[31:30]	RW	保留	2'h0
[29:25]	RW	TMR6_CSR，同 TMR1_CSR	5'h3
[24:20]	RW	TMR5_CSR，同 TMR1_CSR	5'h3
[19:15]	RW	TMR4_CSR，同 TMR1_CSR	5'h3
[14:10]	RW	TMR3_CSR，同 TMR1_CSR	5'h3
[9 : 5]	RW	TMR2_CSR，同 TMR1_CSR	5'h3
[4 : 0]	RW	[4 : 0]为 TMR1_CSR，具体如下：	
		[4]：中断状态寄存器，写 1 清除 1'b0：Timer 无中断产生； 1'b1：Timer 产生中断；	1'b0
		[3]：中断使能寄存器 1'b0：定时时间完成后不产生中断； 1'b1：定时时间完成后产生中断；	1'b0
		[2]：定时器使能寄存器 1'b0：定时器不工作； 1'b1：使能定时器	1'b0
		[1]：定时器工作模式 1'b0：定时器重复定时； 1'b1：定时器只定时一次，定时完成后自动关闭；	1'b1
		[0]：定时器定时单位 1'b0：定时单位为 us； 1'b1：定时单位为 ms；	1'b1

19.4.4 定时器 1 定时值配置寄存器

表 165 定时器 1 定时值配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时器 1 的定时值	32'b0

19.4.5 定时器 2 定时值配置寄存器

表 166 定时器 2 定时值配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时器 2 的定时值	32'b0

19.4.6 定时器 3 定时值配置寄存器

表 167 定时器 3 定时值配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时器 3 的定时值	32'b0

19.4.7 定时器 4 定时值配置寄存器

表 168 定时器 4 定时值配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时器 4 的定时值	32'b0

19.4.8 定时器 5 定时值配置寄存器

表 169 定时器 5 定时值配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时器 5 的定时值	32'b0

19.4.9 定时器 6 定时值配置寄存器

表 170 定时器 6 定时值配置寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时器 6 的定时值	32'b0

19.4.10 定时器 1 当前计数值寄存器

定时器 1 当前计数值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	读取定时器 1 当前计数值;	32'b0

19.4.11 定时器 2 当前计数值寄存器

定时器 2 当前计数值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	读取定时器 2 当前计数值;	32'b0

19.4.12 定时器 3 当前计数值寄存器

定时器 3 当前计数值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	读取定时器 3 当前计数值；	32’b0

19.4.13 定时器 4 当前计数值寄存器

定时器 4 当前计数值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	读取定时器 4 当前计数值；	32’b0

19.4.14 定时器 5 当前计数值寄存器

定时器 5 当前计数值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	读取定时器 5 当前计数值；	32’b0

19.4.15 定时器 6 当前计数值寄存器

定时器 6 当前计数值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	读取定时器 6 当前计数值；	32’b0

20 电源管理模块

20.1 功能概述

PMU 实现芯片硬件工作状态的切换，以及状态切换过程中的电源管理，同时提供定时器、实时时钟以及 32K 时钟。

20.2 主要特性

- 提供芯片电源控制
- 提供定时器功能
- 提供实时时钟控制
- 提供 32K RC 振荡器校准功能
- 提供唤醒功能；

20.3 功能描述

20.3.1 全芯片电源控制

PMU 模块控制芯片的电源开关，包括 40M 起振电路，BandGap，数字 PLL，电压检测电路，数字电路 LDO。

在芯片上电时，PMU 模块根据预设上电顺序引导各模块依次打开电源；

当软件配置寄存器进入休眠模式时，根据安全的下电顺序引导各功能模块依次关闭电源；

当软件配置寄存器进入睡眠模式时，根据顺序关闭时钟，晶体起振电路和相关电源；

在休眠/睡眠模式下提供三种唤醒模式：Timer 定时唤醒，RTC 定时唤醒或者通过将特殊 WAKEUP 管脚拉高唤醒。

20.3.2 低功耗模式

通过配置 PMU 寄存器芯片可以选择 2 种低功耗模式；

Standby 模式：

此模式下数字电源域的电源将被关断，全芯片只有 PMU 模块工作，提供唤醒与复位功能；此时全芯片功耗约 15uA 左右。电源关断后内存中存储数据与内容将全部丢失，唤醒后将重新载入固件，相当于重新启动；

Sleep 模式：

此模式下数字电源域的电源将保留，只是关断 DPLL 与晶体起振电路，切断时钟，此时全芯片功耗约在 1mA 左右；内存中存储数据与代码仍会保留；唤醒后程序将继续运行；

20.3.3 唤醒模式

PMU 支持 3 种唤醒模式，Timer 唤醒，RTC 唤醒和外部 IO 唤醒。

Timer 唤醒

在软件设置休眠/睡眠模式之前，配置 PMU 中 Timer0 模块，设置好休眠时间。当系统进入休眠模式后，当 Timer0 计时到达休眠时间后将会唤醒系统，并产生相应 Timer 中断。系统恢复运行后需要对中断状态寄存器中相应状态位写‘1’清除中断状态，否则,下次进入休眠模式后将立即被中断唤醒；

RTC 唤醒

在软件设置休眠/睡眠模式之前，配置 PMU 中 RTC 模块，设置好休眠时间。当系统进入休眠模式后，当 RTC 记时到达休眠时间后将会唤醒系统，并给出相应 RTC 中断。系统恢复运行后请对中断寄存器 0x14 中相应状态位写‘1’清除中断状态，否则,下次进入休眠模式后将立即被中断唤醒；

外部 IO 唤醒

在软件休眠/睡眠后，PMU 会检测特定 Wakeup 脚，外部控制器可通过将此 IO 拉高来唤醒系统，并给出相应 IO 唤醒中断。PMU 在离开休眠模式后不再检测该 IO 状态。系统恢复运行后请对中断寄存器 0x14 中相应状态位写‘1’清除中断状态，否则,下次进入休眠模式后将立即被中断唤醒；

20.3.4 Timer0 定时器

通过 AHB 寄存器配置定时器使能信号和定时时间。首先设置定时值，然后设置定时器使能 BIT 启动定时器，当达到定时时间后，产生中断，软件通过写清状态寄存器的 BIT0 来清除中断标志。

20.3.5 实时时钟功能

参考实时时钟模块

20.3.6 32K 时钟源切换与校准

W800 芯片集成 32K RC 振荡器作为 PMU 模块时钟来源。

因工作环境与温度变化，32K RC 振荡器的输出频率可能会产生变化，造成计时偏差。因此，在 PMU 模块中引入 32K RC 振荡器校准功能，以及 32K 时钟的切换功能，以便矫正计时偏差。

1) 32K 时钟源的切换

32K 时钟的切换可以通过设置 PS_CR 寄存器的 bit3 为 1，从 32K RC 振荡器切换到由 40M 时钟分频得到的 32K 时钟。但是，当芯片进入休眠模式时，因为 40M 时钟将被关闭，bit3 会自动清 0。待唤醒以后若固件仍要使用精准计时功能，需重新设置 bit3 为 1。

2) 32K RC 振荡电路的校准

首先设置 PS_CR 寄存器的 bit2 为 0，然后再把 PS_CR 寄存器的 bit2 设置为 1。

校准完成后，32K RC 振荡器会相对准确。但是如果希望比较精准的计时，还是建议使用 40M 时钟分频得到的 32K 时钟，此时将得到精确的 RTC 时钟，偏差将只与晶体频偏相关。

20.4 寄存器描述

20.4.1 寄存器列表

表 171 PMU 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	PMU 控制寄存器	PS_CR	RW	用于配置 32K 校准，配置 32K 时钟源，设置芯片的 STANDBY 功能	0X0000_0002
0X0004	PMU 定时器 0	TIMER0	RW	配置定时值（单位为秒），使能定时器	0X0000_0000
0X0008	保留				
0X0014	PMU 中断源寄存器	INT_SRC	RW	提供 PMU 中断标志	0X0000_0000

20.4.2 PMU 控制寄存器

表 172 PMU 控制寄存器

位	访问	操作说明	复位值

[31: 11]	RO		24'b0
[10]	RW	唤醒按键中断极性选择： 0： 唤醒按键为高电平时置位中断； 1： 唤醒按键为高电平时置位中断； 只在 Sleep 中断有效；	1'b0
[9:6]	RW	按键唤醒最低保持时长数： 单位：128ms；	4'd01
[5]	RW	DLDO_CORE 参考电压源选择 1: ABG 0: DBG	1'b1
[4]	RW	32K 振荡电路 BYPASS 信号 高有效，置位为 1 后 32K 为 40M 时钟分频得到。②	1'b0
[3]	RW	RC 32K 振荡器校准电路启动开关； 1'b0：校准电路复位； 1'b1：启动校准电路； 要启动校准功能，需要此位先置 0，后置 1。	1'b0
[2]	RW	使能按键触发进入睡眠功能 0：不使能； 1：在 active 模式时按下 io_wakeup 按键达到到阈值时间，将触发 io_sleep_flag 中断，上报给 MCU。	1'b0
[1]	RW	Sleep 使能信号，高有效。 1'b0：芯片唤醒状态	1'b1

		1’b1：芯片进入 Sleep 状态 如果 WAKEUP 脚为无效电平，且没有配置 TIMER0/1 中断唤醒，则该寄存器有效时，芯片进入 Sleep 状态； 如果唤醒中断产生，则芯片会从 Sleep 状态切换到唤醒状态，唤醒条件满足，该位自动清 0。 **唤醒源：WAKEUP 脚，TIMER0/TIMER1，RTC** 1） WAKEUP 脚，高有效；要想芯片进入 Sleep 状态，WAKEUP 脚必须处于低电平。要唤醒时，拉高 WAKEUP 脚，产生唤醒中断，使芯片离开 Sleep 状态。 2） TIMER0，定时器唤醒中断。 当 WAKEUP 脚为低，TIMER0 设置定时时间并使能，定时时间到会产生唤醒中断，使芯片离开 Sleep 状态。 3） RTC，定时时间到唤醒 当 WAKEUP 脚为低，RTC 定时时间到，会产生唤醒中断，使芯片离开 Sleep 状态	
[0]	RW	STANDBY 使能信号，高有效。 1’b0：芯片唤醒状态 1’b1：芯片进入 STANDBY 状态 如果 WAKEUP 脚为无效电平，且没有配置 TIMER0/1 中断唤醒，则该寄存器有效时，芯片进入 STANDBY 状态； 如果唤醒中断产生，则芯片会从 STANDBY 状态切换到唤醒状态，唤醒条件满足，该位自动清 0。	1’b0

		唤醒源：WAKEUP 脚，TIMER0/TIMER1，RTC 4） WAKEUP 脚，高有效；要想芯片进入 STANDBY 状态，WAKEUP 脚必须处于低电平。要唤醒时，拉高 WAKEUP 脚，产生唤醒中断，使芯片离开 STANDBY 状态。 5） TIMER0，定时器唤醒中断。 当 WAKEUP 脚为低，TIMER0 设置定时时间并使能，定时时间到会产生唤醒中断，使芯片离开 STANDBY 状态。 6） RTC，定时时间到唤醒 当 WAKEUP 脚为低，RTC 定时时间到，会产生唤醒中断，使芯片离开 STANDBY 状态	

20.4.3 PMU 定时器 0

表 173 PMU 定时器 0 寄存器

位	访问	操作说明	复位值
[31:17]	RO	保留	15’b0
[16]	RW	Timer0 使能位 1’b0：位使能。 1’b1：使能；	1’b0
[15: 0]	RW	Timer0 的定时值，单位:秒	16’b0

20.4.4 PMU 中断源寄存器

表 174 PMU 中断源寄存器

位	访问	操作说明	复位值
[31: 9]	R	保留	
[8]	RW	显示当次上电状态： 1'b0：上电或复位启动 1'b1：从休眠状态唤醒，写 1 清除	1'b0
[7]	RO	保留	1'b0
[6]	RO	保留	1'b0
[5]	RW	RTC 定时中断标志位： 1'b0：有定时中断产生 1'b1：无定时中断产生，写 1 清除	1'b0
[4]	RW	保留	1'b0
[3]	RW	保留	1'b0
[2]	RW	WAKEUP 管脚唤醒中断标志位 1'b0：无 WAKEUP 唤醒中断产生 1'b1：有 WAEKUP 唤醒中断产生，写 1 清除	1'b0
[1]	RW	保留	1'b0
[0]	RW	Timer0 定时中断标志位： 1'b0：无 Timer0 中断产生 1'b1：有 Timer0 中断产生，写 1 清除	1'b0

21 实时时钟模块

21.1 功能概述

RTC 是由 PMU 模块提供的 BCD 计数器/定时器，两个 32 位寄存器包含秒、分、时、日、月、年，以二进制编码的十进制格式表示（BCD），能自动对 28、29（闰年）、30、31 天的月份进行修正。

在相应软件配置下，RTC 既可以提供时钟日历功能，又可以当作定时器使用，在定时器达到设置的时间后会产生一个 RTC 中断，可用来唤醒处于睡眠状态的系统。

RTC 模块有两个时钟源可以配置：40M 时钟分频和内部 32K 时钟。正常工作时可由软件配置具体使用哪个时钟源；睡眠状态时只能使用 32K 时钟。如果正常工作状态 RTC 时钟源由 40M 时钟分频所得，那么进入睡眠状态后会自动切换到 32K 时钟，系统被唤醒以后仍然保持使用 32K 时钟。所以只要电源电压保持在工作范围内，无论模块是正常工作状态还是睡眠状态，RTC 模块都不会停止工作。

21.2 主要特性

- 提供计时功能
- 提供定时功能
- 提供定时中断
- 中断唤醒系统

21.3 功能描述

21.3.1 计时功能

在 RTC 配置寄存器 1 中可配置日、时、分、秒初始值，在 RTC 配置寄存器 2 中可配置年、月初始值，在 RTC 配置寄存器 2 可使能计时功能。

在 RTC 计时功能使能之后，读取 RTC 配置寄存器 1 可得到当前的日、时、分、秒数值，读取 RTC 配置寄

存器 2 可得到当前的年、月数值。

21.3.2 定时功能

在 RTC 配置寄存器 1 中可配置日、时、分、秒定时值，在 RTC 配置寄存器 2 中可配置年、月定时值，在 RTC 配置寄存器 1 可使能定时功能。

当 RTC 定时器到达定时时间后会产生一个 RTC 中断，此时设置 PMU 中断源寄存器 RTC 中断位为 1 可清除中断状态。

当系统进入睡眠模式之后，RTC 定时器产生的中断会唤醒系统。

21.4 寄存器描述

21.4.1 寄存器列表

RTC 模块一共有 2 个 32 位专属寄存器，RTC 中断状态需要查询 PMU 中断源寄存器。

表 175 RTC 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X000C	RTC 配置寄存器 1	RTC_R1	RW	配置 RTC 日时分秒值，配置使能定时	0X0000_0000
0X0010	RTC 配置寄存器 2	RTC_R2	RW	配置 RTC 年月值，配置使能计时	0X0000_0000

21.4.2 RTC 配置寄存器 1

表 176 RTC 配置寄存器 1

位	访问	操作说明	复位值
[31]	RW	RTC 定时中断功能使能	1'b0

		1'b0：不使能 1'b1：使能	
[30:29]		保留	
[28:24]	RW	日初值/日定时值	5'b0
[23:21]		保留	
[20:16]	RW	小时初值/小时定时值	5'b0
[15:14]		保留	
[13: 8]		保留	
[7 : 6]		保留	
[5 : 0]	RW	秒初值/秒定时值	6'b0

21.4.3 RTC 配置寄存器 2

表 177 RTC 配置寄存器 2

位	访问	操作说明	复位值
[31:17]		保留	
[16]	RW	RTC 计时功能使能位 1'b0：不使能 1'b1：使能	1'b0
[15]		保留	
[14: 8]	RW	年初值/年定时值	7'b0
[7 : 4]		保留	
[3 : 0]	RW	月初值/月定时值	4'b0

22 看门狗模块

22.1 功能概述

实现“看门狗”功能。设计用于系统崩溃时全局复位。

“看门狗”会产生一个周期性中断，系统软件在中断产生后要清除其中断标志，若超过其设定时间未清除，则会产生一个硬复位信号对系统进行复位。

22.2 主要特性

- 提供定时功能
- 提供复位功能
- 提供定时中断

22.3 功能描述

22.3.1 定时功能

设置定时值到寄存器 WD_LD 后，设置 WDG_CTRL 的 BIT0 为 1 启动定时器，WDG 模块定时时间到会产生定时中断，通知程序处理。如果寄存器 WD_CLR 的 BIT0 不清除，则会周期产生定时中断。

WD_LD 的值以 APB 时钟单位为基准，APB 的时钟从 160M 时钟分频出来。

22.3.2 复位功能

设置芯片定时值 WD_LD 后，启动定时和复位功能（设置 WDG_CTRL 的 BIT1/BIT0），WDG 模块启动倒计时，定时时间到，WDG 会产生定时中断，同时如果 WD_CLR 的 BIT0 不清除，则芯片会在定时时间的下一个周期产生复位信号。

22.4 寄存器描述

22.4.1 寄存器列表

表 178 WDG 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	WDG 定时加载寄存器	WD_LD	RW	配置定时值，用于重复加载	0XFFFF_FFFF
0X0004	WDG 当前值寄存器	WD_VAL	RO	获取当前定时器的值	0XFFFF_FFFF
0X0008	WDG 控制寄存器	WD_CTRL	RW	控制寄存器	0X0000_0000
0X000C	WDG 中断清除寄存器	WD_CLR	WO	中断清除寄存器	0X0000_0000
0X0010	WDG 中断源寄存器	WD_SRC	RO	中断源寄存器	0X0000_0000
0X0014	WDG 中断输出寄存器	WD_STATE	RO	中断输出状态寄存器	0X0000_0000

22.4.2 WDG 定时值加载寄存器

表 179 WDG 定时值加载寄存器

位	访问	操作说明	复位值
[31: 0]	RW	配置定时值，用于重复加载 此寄存器的值以 APB 时钟为计数单位。 例如：以 APB 时钟为 40MHZ，则定时值最大时长约 107s 左右，即 0FFFFFFFF/40000000	32'hffff_ffff

22.4.3 WDG 当前值寄存器

表 180 WDG 当前值寄存器

位	访问	操作说明	复位值
[31: 0]	RO	获取当前定时器的值 要计算剩余时间，只要读取当前值即可。 要计算已经过去的时间，只要把寄存器 WD_LD 的值减去寄存器 WD_VAL 的值即可	32'hffff_ffff

22.4.4 WDG 控制寄存器

表 181 WDG 控制寄存器

位	访问	操作说明	复位值
[31: 2]		保留	30'h0
[1]	RW	复位使能位 1'b0：WDG 复位条件产生时，不产生复位信号 1'b1：WDG 复位条件产生时，产生复位信号	1'b0
[0]	RW	定时使能位 1'b0：定时器不工作 1'b1：定时器工作，产生周期性中断	1'b0

22.4.5 WDG 中断清除寄存器

表 182 WDG 中断清除寄存器

位	访问	操作说明	复位值
[31: 1]		保留	31'h0

[0]	WO	中断状态清除位，写任意值清除当前中断状态	1'b0

22.4.6 WDG 中断源寄存器

表 183 WDG 中断源寄存器

位	访问	操作说明	复位值
[31: 1]		保留	31'h0
[0]	RO	中断源寄存器，定时器功能打开，会同时产生该中断	1'b0

22.4.7 WDG 中断状态寄存器

表 184 WDG 中断状态寄存器

位	访问	操作说明	复位值
[31: 1]		保留	31'h0
[0]	RO	中断输出状态寄存器。该中断在定时器关闭后不产生，但 WD_SRC 可能为 1	1'b0

23 PWM 控制器

23.1 功能概述

PWM 是一种对模拟信号电平进行数字编码的方法。通过高分辨率计数器的使用，方波的占空比被调制用来对一个具体模拟信号的电平进行编码。PWM 信号仍然是数字的，因为在给定的任何时刻，满幅值的直流供电要么完全有（ON），要么完全无（OFF）。电压或电流源是以一种通（ON）或断（OFF）的重复脉冲序列被加到模拟负载上去的。通的时候即是直流供电被加到负载上的时候，断的时候即是供电被断开的时候。只要带宽足够，任何模拟值都可以使用 PWM 进行编码。

23.2 主要特性

- 支持 2 通道输入信号捕获功能（PWM0 和 PWM4 两个通道）
- 输入信号捕获功能支持中断交互模式和 DMA 传输模式；DMA 模式支持按字操作
- 支持 5 通道 PWM 信号生成功能
- 5 通道 PWM 信号生成支持单次生成模式和自动装载模式
- 支持 5 通道制动功能
- PWM 输出频率范围：3Hz~160kHz
- 占空比最大精度：1/256，插入死区的计数器宽度：8bit
- 支持通道 0 通道 1 同步功能，支持通道 2 通道 3 同步功能
- 支持通道 0 通道 1 的互补与非互补模式，支持通道 2 通道 3 的互补与非互补模式
- 支持 5 通道同步功能

23.3 功能描述

23.3.1 输入信号捕获

PWM 控制器支持两个通道的信号捕获功能，通过设置 PWM_CTL 寄存器的 Bit24 可以激活通道 0 的捕获功能，通过设置 PWM_CAP2CTL 寄存器的 Bit1 可以激活通道 4 的捕获功能。对捕获的信号的电平还可以设置是否翻转功能。通道捕获到相应的信号后，捕获数在更新到相应的捕获寄存器 PWM_CAPDAT(通道 0 捕获数)和 PWM_CAP2DAT(通道 4 捕获数)。

23.3.2 DMA 传输捕获数

通道 0 或者通道 4 开启捕获功能后，捕获寄存器的计数可以通过 DMA 通道快速传输至内存，加速用户处理进程。

23.3.3 支持单次和自动装载模式

PWM 控制器的五路输出通道均支持单次输出模式和自动装载模式。单次装载模式下，通道输出指定周期个波形后，就不再输出 PWM 波了；自动装载模式下，通道输出指定周期个波形后，会自动重新装载周期数，从而继续产生 PWM 波。

23.3.4 多种输出模式

PWM 控制器支持独立输出模式，即每个通道独立输出，互不干预；支持双通道同步模式，即一个通道的输出完全与另一个通道输出一致；支持五通道同步模式，通道 1 至通道 4 的输出完全与通道 0 的输出一致；支持双通道互补输出，即一个通道输出的波形与另一个通道输出的波形完全相反；支持互补模式下常会用到的死区设置，死区长度最多可设置 256 个时钟周期；支持制动模式，当制动端口检测到指定电平后，输出通道会输出已经设置好的制动电平。

多种输出模式灵活可配，满足了使用者对 PWM 相关的各种应用场景。

23.4 寄存器描述

23.4.1 PWM 寄存器列表

表 185 PWM 寄存器列表

偏移地址	名称	缩写	访问	描述	复位值
0X0000	时钟分频寄存器_01	PWM_CLKDIV01	RW	对通道 0 和通道 1 的时钟进行分频	0X0000_0000
0X0004	时钟分频寄存器_23	PWM_CLKDIV23	RW	对通道 2 和通道 3 的时钟进行分频	0X0000_0000
0X0008	控制寄存器	PWM_CTL	RW	用以配置或者控制一些可配置项	0X0000_0000
0X000C	周期寄存器	PWM_PERIOD	RW	用以设置通道 0 至通道 4 的周期	0X0000_0000
0X0010	周期数寄存器	PWM_PNUM	RW	用以设置通道 0 至通道 4 的信号生成周期数	0X0000_0000
0X0014	比较寄存器	PWM_CMPDAT	RW	用以存放通道 0 至通道 4 的比较值以产生不同的占空比	0X0000_0000
0X0018	死区控制寄存器	PWM_DTCTL	RW	用以配置或者控制死区相关的可配置项	0X0000_0000
0X001C	中断控制寄存器	PWM_INTEN	RW	用以对相关中断进行使能控制	0X0000_0000
0X0020	中断状态寄存器	PWM_INTSTS	RW	用以查询相关中断的状态	0X0000_0000
0X0024	通道 0 捕获寄存器	PWM_CAPDAT	RO	用以捕获并计数来到通道 0 的上升沿和下降沿	0X0000_0000
0X0028	制动控制寄存器	PWM_BRKCTL	RW	用以对制动模式进行控制	0X0000_0000
0X002C	时钟分频寄存器_4	PWM_CH4_reg1	RW	对通道 4 的时钟进行分频	0X0000_0000

0X0030	通道 4 控制寄存器_1	PWM_CH4_reg2	RW	对通道 4 的相关配置项进行设置	0X0000_0000
0X0034	通道 4 捕获寄存器	PWM_CAP2DAT	RO	用以捕获并计数来到通道 4 的上升沿和下降沿	0X0000_0000
0X0038	通道 4 控制寄存器_2	PWM_CAP2CTL	RW	通道 4 的相关配置项进行设置	0X0000_0000

23.4.2 时钟分频寄存器_01

表 186 PWM 时钟分频寄存器_01

位	访问	操作说明	复位值
[31:16]	RW	CLKDIV1 CH1 分频计数器 由计数器值决定分频数 注意：分频范围为(0~65535)，如不需要分频，输入 0 或 1。	16'h0
[15:0]	RW	CLKDIV0 CH0 分频计数器 同 CH1	16'h0

23.4.3 时钟分频寄存器_23

表 187 PWM 时钟分频寄存器_23

位	访问	操作说明	复位值
[31:16]	RW	CLKDIV3 CH3 分频计数器 同 CH1	16'h0
[15:0]	RW	CLKDIV2	16'h0

		CH2 分频计数器 同 CH1	

23.4.4 控制寄存器

表 188 PWM 控制寄存器

位	访问	操作说明	复位值
[31:27]	RW	CNTEN 计数器计数使能 1'b0：停止计数 1'b1：开始计数 注意：每位分别控制每个通道，从高到低依次控制 CH4、CH3、CH2、CH1 和 CH0	5'b0
[26]	--	保留	1'b0
[25]	RW	CAPINV 捕获反向使能标识位 1'b0：捕获模式输入信号反向无效 1'b1：捕获模式输入信号反向有效，对输入信号取反	1'b0
[24]	RW	CPEN 捕获功能使能标识位 1'b0：CH0 捕获功能无效，RCAPDAT 和 FCAPDAT 值不会被更新； 1'b1：CH0 捕获功能有效，捕获并锁存 PWM 计数器，分别存储在 RCAPDAT（上升沿锁存）和 FCAPDAT（下降沿锁存）	1'b0
[23:22]	RW	CNTTYPE3	2'b0

		CH3 计数器计数方式 2'b00：边缘对齐模式（计数器计数方式为递增，仅针对捕获模式） 2'b01：边缘对齐模式（计数器计数方式为递减，仅针对 PWM 模式） 2'b10：中央对齐模式（仅针对 PWM 模式） 注意：在 PWM 模式下，当计数器被设置为沿对齐模式时，需要设置计数方式为递减方式。	
[21:20]	RW	CNTTYPE2 CH2 计数器计数方式 同 CH3	2'b0
[19:18]	RW	CNTTYPE1 CH1 计数器计数方式 同 CH3	2'b0
[17:16]	RW	CNTTYPE0 CH0 计数器计数方式 同 CH3	2'b0
[15:14]	RW	TWOSYNCEN 2 通道同步模式使能信号 1'b0：不允许 2 通道同步 1'b1：允许 2 通道同步， PWM_CH0 和 PWM_CH1 具有相同的相位，且相位由 PWM_CH0 决定；PWM_CH2 和 PWM_CH3 具有相同的相位，且相位由 PWM_CH2 决定 15bit 控制 CH3 和 CH2	2'b0

		14bit 控制 CH1 和 CH0	
[13]	--	保留	1'b0
[12]	RW	POEN PWM 管脚输出使能位 1'b0：PWM 管脚置为输出状态 1'b1：PWM 管脚置为三态状态 注意：只是针对 CH0	1'b0
[11:8]	RW	CNTMODE PWM 生成循环方式 1'b0：单次模式 1'b1：自动装载模式 注意：CNTMODE 变化过程中，PWM_CMPDAT 归零；每位分别控制每个通道，从高到低依次控制 PW3、PW2、PW1 和 PW0	4'h0
[7]	--	保留	1'b0
[6]	RW	ALLSYNCEN 全通道同步模式使能信号 1'b0：不允许全部通道同步 1'b1：允许全部通道同步，PWM_CH0、PWM_CH1、PWM_CH2 和 PWM_CH3 具有相同的相位，且相位由 PWM_CH0 决定	1'b0
[5:2]	RW	PINV PWM 输出信号极性使能 1'b0：PWM 输出极性翻转不使能	4'h0

		1’b1：PWM 输出极性翻转使能 注意：每位分别控制每个通道，从高到低依次控制 PW3、PW2、PW1 和 PW0	
[1:0]	RW	OUTMODE 输出模式 1’b0：每两个通道非互补模式 1’b1：每两个通道组成互补模式 BIT1 控制 CH2 和 CH3 BIT0 控制 CH0 和 CH1	2’b0

23.4.5 周期寄存器

表 189 PWM 周期寄存器

位	访问	操作说明	复位值
[31:24]	RW	PERIOD3 CH3 周期寄存器值(注意：period 不可以大于 255) “沿对齐模式（计数器计数方式为递减）”： ➢ PERIOD 寄存器值，周期值为（PERIOD + 1） ➢ 占空比=（CMP+1）/（PERIOD + 1） ➢ CMP>=PERIOD：PWM 输出固定为高 ➢ CMP<PERIOD：PWM 低电平宽度为（PERIOD-CMP），高电平宽度为（CMP+1） ➢ CMP=0：PWM 低电平宽度为 PERIOD，高电平宽度为 1； “中间对齐模式”： ➢ PERIOD 寄存器值：周期为 2*（PERIOD+1）	8’h0

		➢ 占空比=(2*CMP+1）/（2*（PERIOD+1)） ➢ CMP>PERIOD：PWM 持续为高 ➢ CMP<=PERIOD：PWM 低电平=2*（PERIOD-CMP）+1， 高电平=（2*CMP）+1 ➢ CMP=0：PWM 低电平宽度为 2*PERIOD+1，高电平宽度为 1。 注意：“中间对齐模式”中，周期数不应为 255。 无论选择哪种对齐模式，通道周期均由分频数（N）和周期数（P）共同决定，即：输入时钟为 40MHz，经分频后时钟频率 f_div 为：f_div = 40MHz/N，N 为分频数（16bit)。输出频率 f_output 为：f_output = f_div / P，P 为周期数。 注意：在 PWM 模式下，当计数器被设置为沿对齐模式时，需要设置计数方式为递减方式。	
[23:16]	RW	PERIOD2 CH2 周期寄存器值(注意：period 不可以大于 255) 同 PERIOD3	8’h0
[15:8]	RW	PERIOD1 CH1 周期寄存器值(注意：period 不可以大于 255) 同 PERIOD3	8’h0
[7:0]	RW	PERIOD0 CH0 周期寄存器值(注意：period 不可以大于 255) 同 PERIOD3	8’h0

23.4.6 周期数寄存器

表 190 PWM 周期数寄存器

位	访问	操作说明	复位值
[31:24]	RW	PNUM3 PWM3 生成周期数 设置 PWM3 周期数 PNUM3，当 PWM 产生 PNUM3 个 PWM 信号后，停止生成信号，同时触发中断和置位中断状态字	8'h0
[23:16]	RW	PNUM2 PWM2 生成周期数 同 PNUM3	8'h0
[15:8]	RW	PNUM1 PWM1 生成周期数 同 PNUM3	8'h0
[7:0]	RW	PNUM0 PWM0 生成周期数 同 PNUM3	8'h0

23.4.7 比较寄存器

表 191 PWM 比较寄存器

位	访问	操作说明	复位值
[31:24]	RW	CMP3 PWM3 比较寄存器值	8'h0

		“沿对齐模式（计数器计数方式为递减）”： ➢ PERIOD 寄存器值，周期值为（PERIOD + 1） ➢ 占空比=（CMP+1）/（PERIOD + 1） ➢ CMP>=PERIOD：PWM 输出固定位高 ➢ CMP<PERIOD：PWM 低电平宽度为（PERIOD-CMP），高电平宽度为（CMP+1） ➢ CMP=0：PWM 低电平宽度为 PERIOD，高电平宽度为 1； “中间对齐模式”： ➢ PERIOD 寄存器值：周期为 2*（PERIOD+1） ➢ 占空比=(2*CMP+1）/2*（PERIOD+1） ➢ CMP>PERIOD：PWM 持续为高 ➢ CMP<=PERIOD：PWM 低电平=2*（PERIOD-CMP）+1，高电平=（2*CMP）+1 ➢ CMP=0：PWM 低电平宽度为 2*PERIOD+1，高电平宽度为 1。 无论选择哪种对齐模式，通道周期均由分频数（N）和周期数（P）共同决定，即： 输入时钟为 40MHz，经分频后时钟频率 f_div 为：f_div = 40MHz/N，N 为分频数（16bit）。输出频率 f_output 为：f_output = f_div / P，P 为周期数。 注意：在 PWM 模式下，当计数器被设置为沿对齐模式时，需要设置计数方式为递减方式。	
[23:16]	RW	CMP2 PWM2 比较寄存器值 同 CMP3	8’h0
[15:8]	RW	CMP1 PWM1 比较寄存器值	8’h0

		同 CMP3	
[7:0]	RW	CMP0 PWM0 比较寄存器值 同 CMP3	8'h0

23.4.8 死区控制寄存器

表 192 PWM 死区控制寄存器

位	访问	操作说明	复位值
[31:22]	--	保留	10'h0
[21]	RW	DTEN23 通道 2 和通道 3 是否可以插入死区有效标识 插入死区有效信号只有在通道的互补模式打开后，才有效。并且，如果插入有效信号为 0，则两个通道输出的互补信号没有死区的插入 1'b0：插入死区无效 1'b1：插入死区有效	1'b0
[20]	RW	DTEN01 通道 0 和通道 1 是否可以插入死区有效标识 同 DTEN23	1'b0
[19:18]	--	保留	2'b0
[17:16]	RW	DTDIV 死区时钟分频控制 2'b00：死区时钟等于基准时钟（40MHz）	2'b0

		2'b01：死区时钟等于基准时钟（40MHz）二分频 2'b10：死区时钟等于基准时钟（40MHz）四分频 2'b11：死区时钟等于基准时钟（40MHz）八分频	
[15:8]	RW	DTCNT23 通道 3 和通道 2 的死区间隔 8bit 决定死区间隔值，死区时钟由 DTDIV 决定	8'h0
[7:0]	RW	DTCNT01 通道 1 和通道 0 的死区间隔 8bit 决定死区间隔值，死区时钟由 DTDIV 决定	8'h0

23.4.9 中断控制寄存器

表 193 PWM 中断控制寄存器

位	访问	操作说明	复位值
[31:8]	--	保留	24'h0
[7]	RW	DMA_request_EN DMA_request 使能 1'b0：DMA_request 无效 1'b1：DMA_request 有效	1'b0
[6]	RW	FLIEN 下降沿缓存中断使能位 1'b0：下降沿缓存中断无效 1'b1：下降沿缓存中断有效	1'b0

		注意：针对 CH0 而言	
[5]	RW	RLIEN 上升沿缓存中断使能位 1'b0：上升缓存中断无效 1'b1：上升沿缓存中断有效 注意：针对 CH0 而言	1'b0
[4:0]	RW	PIEN PWM 周期中断使能位 1'b0：周期中断无效 1'b1：周期中断有效 注意：当计数器计数到 0，且 PWM 周期个数满足 PWM_PNUM 后，触发中断。	5'b0

23.4.10 中断状态寄存器

表 194 PWM 中断状态寄存器

位	访问	操作说明	复位值
[31:10]	--	保留	12'h0
[9]	RW	OVERFL 计数器溢出标志 1'b0：捕获模式，计数器计数过程中，计数器未溢出 1'b1：捕获模式，计数器计数过程中，计数器溢出 注意：当用户清除 CFLIF 或 CRLIF 时，本 bit 也同时被清除	1'b0
[8]	RW	FLIFOV	1'b0

		下降沿延迟中断标识过跑状态 1'b0：当 CFILF 为 1 时，无下降沿延迟中断产生 1'b1：当 CFILF 为 1 时，又一次发生下降沿延迟中断 注意：当用户清除 CFILF 时，本 bit 也同时被清除	
[7]	RW	RLIFOV 上升沿延迟中断标识过跑状态 1'b0：当 CRILF 为 1 时，无上升沿延迟中断产生 1'b1：当 CRILF 为 1 时，又一次发生上升沿延迟中断 注意：当用户清除 CRILF 时，本 bit 也同时被清除	1'b0
[6]	RW	CFLIF 捕获下降沿中断标识 1'b0：没有捕获到下降沿 1'b1：当捕获到下降沿，本位被设置为 1 注意：通过写入 1，清除该标识位； 注意：针对 CH0 而言	1'b0
[5]	RW	CRLIF 捕获上升沿中断标识 1'b0：没有捕获到上升沿 1'b1：当捕获到上升沿，本位被设置为 1 注意：通过写入 1，清除该标识位； 注意：针对 CH0 而言	1'b0
[4:0]	RW	PIF	5'b0

		PWM 周期中断标识 当 PWM 产生指定周期 PWM 信号后，该标识位置 1；通过软件写入 1，清除该标识 注意：每位分别标识每个通道，从高到低依次控制 PW4、PW3、PW2、PW1 和 PW0	

23.4.11 通道 0 捕获寄存器

表 195 PWM 通道 0 捕获寄存器

位	访问	操作说明	复位值
[31:16]	RO	PWM_FCAPDAT 捕获下降沿寄存器 当输入信号存在下降沿时，存储当前计数器值	16'h0
[15:0]	RO	PWM_RCAPDAT 捕获上升沿寄存器 当输入信号存在上升沿时，存储当前计数器值	16'h0

23.4.12 制动控制寄存器

表 196 PWM 制动控制寄存器

位	访问	操作说明	复位值
[31:16]	--	保留	16'h0
[15:11]	RW	BRKCTL 制动模式使能 1'b0：制动模式禁止 1'b1：制动模式启动	5'b0

		[7:3]分别对应 CH4、CH3、CH2、CH1 和 CH0	
[10:8]	--	保留	3'b0
[7:3]	RW	BKOD 制动输出控制寄存器 1'b0：当制动模式有效时，PWM 输出低电平 1'b1：当制动模式有效时，PWM 输出高电平 [7:3]分别对应 CH4、CH3、CH2、CH1 和 CH0	5'b0
[2:0]	--	保留	3'b0

23.4.13 时钟分频寄存器_4

表 197 PWM 时钟分频寄存器_4

位	访问	操作说明	复位值
[31:16]	RW	CLKDIV4 CH4 分频计数器 由计数器值决定分频数 注意：分频范围为(0~65535)，如不需要分频，输入 0 或 1。	16'h0
[15:8]	RW	PERIOD4 CH4 周期寄存器值(注意：period 不可以大于 255) “沿对齐模式（计数器计数方式为递减）”： ➢ PERIOD 寄存器值，周期值为（PERIOD + 1） ➢ 占空比=（CMP+1）/（PERIOD + 1） ➢ CMP>=PERIOD：PWM 输出固定位高	8'h0

		➢ CMP<PERIOD：PWM 低电平宽度为（PERIOD-CMP），高电平宽度为（CMP+1） ➢ CMP=0：PWM 低电平宽度为 PERIOD，高电平宽度为 1； “中间对齐模式”： ➢ PERIOD 寄存器值：周期为 2*（PERIOD+1） ➢ 占空比=(2*CMP+1）/（2*（PERIOD+1）） ➢ CMP>PERIOD：PWM 持续为高 ➢ CMP<=PERIOD：PWM 低电平=2*（PERIOD-CMP）+1，高电平=（2*CMP）+1 ➢ CMP=0：PWM 低电平宽度为 2*PERIOD+1，高电平宽度为 1。 注意：“中间对齐模式”中，周期数不应为 255。 无论选择哪种对齐模式，通道周期均由分频数（N）和周期数（P）共同决定，即：输入时钟为 40MHz，经分频后时钟频率 f_div 为：f_div = 40MHz/N，N 为分频数（16bit）。输出频率 f_output 为：f_output = f_div / P，P 为周期数。 **注意：**在 PWM 模式下，当计数器被设置为沿对齐模式时，需要设置计数方式为递减方式。	
[7:0]	RW	CH4 生成周期数 设置 PWM4 周期数为 PNUM4，当 PWM 产生 PNUM4 个 PWM 信号后，停止生成信号，同时触发中断和置位中断状态字	8’h0

23.4.14 通道 4 控制寄存器_1

表 198 PWM 通道 4 控制寄存器_1

位	访问	操作说明	复位值
[31:16]	--	保留	16’h0

[15:8]	RW	CMP4 CH4 周期寄存器值 “沿对齐模式（计数器计数方式为递减）”： ➢ PERIOD 寄存器值，周期值为（PERIOD + 1） ➢ 占空比=（CMP+1）/（PERIOD + 1） ➢ CMP>=PERIOD：PWM 输出固定位高 ➢ CMP<PERIOD：PWM 低电平宽度为（PERIOD-CMP），高电平宽度为（CMP+1） ➢ CMP=0：PWM 低电平宽度为 PERIOD，高电平宽度为 1； “中间对齐模式”： ➢ PERIOD 寄存器值：周期为 2*（PERIOD+1） ➢ 占空比=(2*CMP+1）/2*（PERIOD+1） ➢ CMP>PERIOD：PWM 持续为高 ➢ CMP<=PERIOD：PWM 低电平=2*（PERIOD-CMP）+1，高电平=（2*CMP）+1 ➢ CMP=0：PWM 低电平宽度为 2*PERIOD+1，高电平宽度为 1。 无论选择哪种对齐模式，通道周期均由分频数（N）和周期数（P）共同决定，即：输入时钟为 40MHz，经分频后时钟频率 f_div 为：f_div = 40MHz/N，N 为分频数（16bit）。输出频率 f_output 为：f_output = f_div / P，P 为周期数。 **注意：**在 PWM 模式下，当计数器被设置为沿对齐模式时，需要设置计数方式为递减方式。	8’h0
[7:5]	--	保留	3’b0
[4:3]	RW	CNTTYPE4 CH4 计数器计数方式	2’b0

		2’b00：边缘对齐模式（计数器计数方式为递增，仅针对捕获模式） 2’b01：边缘对齐模式（计数器计数方式为递减，仅针对 PWM 模式） 2’b10：中央对齐模式（仅针对 PWM 模式） **注意：**在 PWM 模式下，当计数器被设置为沿对齐模式时，需要设置计数方式为递减方式。	
[2]	--	保留	1’b0
[1]	RW	CNTMODE4 CH4 生成循环方式 1’b0：单次模式 1’b1：自动装载模式 注意：CNTMODE 变化过程中，PWM_CMPDAT 归零	1’b0
[0]	RW	PINV4 CH4 输出信号极性使能 1’b0：PWM 输出极性翻转不使能 1’b1：PWM 输出极性翻转使能	1’b0

23.4.15 通道 4 捕获寄存器

表 199 PWM 道 4 捕获寄存器

位	访问	操作说明	复位值
[31:16]	RO	PWM_FCAP2DAT 捕获下降沿寄存器 当输入信号存在下降沿时，存储当前计数器值	16’h0

[15:0]	RO	PWM_RCAP2DAT 捕获上升沿寄存器 当输入信号存在上升沿时，存储当前计数器值	16'h0

23.4.16 通道 4 控制寄存器_2

表 200 PWM 通道 4 控制寄存器_2

位	访问	操作说明	复位值
[31:11]	--	保留	21'h0
[10]	RW	DMA_request2_mask DMA_request2 使能 1'b0：DMA_request2 无效 1'b1：DMA_request2 有效 注意：只是针对 CH4	1'b0
[9]	RW	FLIEN2 下降沿缓存中断使能位 1'b0：下降沿缓存中断无效 1'b1：下降沿缓存中断有效 注意：只是针对 CH4	1'b0
[8]	RW	RLIEN2 上升沿缓存中断使能位 1'b0：上升缓存中断无效 1'b1：上升沿缓存中断有效 注意：只是针对 CH4	1'b0

[7]	RW	OVERFL2 计数器溢出标志 1'b0：捕获模式，计数器计数过程中，计数器未溢出 1'b1：捕获模式，计数器计数过程中，计数器溢出 注意：当用户清除 CFLIF 或 CRLIF 时，本 bit 也同时被清除 注意：只是针对 CH4	1'b0
[6]	RW	FLIFOV2 下降沿延迟中断标识过跑状态 1'b0：当 CFILF 为 1 时，无下降沿延迟中断产生 1'b1：当 CFILF 为 1 时，又一次发生下降沿延迟中断 注意：当用户清除 CFILF 时，本 bit 也同时被清除 注意：只是针对 CH4	1'b0
[5]	RW	RLIFOV2 上升沿延迟中断标识过跑状态 1'b0：当 CRILF 为 1 时，无上升沿延迟中断产生 1'b1：当 CRILF 为 1 时，又一次发生上升沿延迟中断 注意：当用户清除 CRILF 时，本 bit 也同时被清除 注意：只是针对 CH4	1'b0
[4]	RW	CFLIF2 捕获下降沿中断标识 1'b0：没有捕获到下降沿 1'b1：当捕获到下降沿，本位被设置为 1	1'b0

		注意：通过写入 1，清除该标识位 注意：只是针对 CH4	
[3]	RW	CRLIF2 捕获上升沿中断标识 1'b0：没有捕获到上升沿 1'b1：当捕获到上升沿，本位被设置为 1 注意：通过写入 1，清除该标识位 注意：只是针对 CH4	1'b0
[2]	RW	POEN2 PWM 管脚输出使能位 1'b0：PWM 管脚置为输出状态 1'b1：PWM 管脚置为三态状态 注意：只是针对 CH4	1'b0
[1]	RW	CPEN2 捕获功能使能标识位 1'b0：CH4 捕获功能无效，RCAPDAT 和 FCAPDAT 值不会被更新； 1'b1：CH4 捕获功能有效，捕获并锁存 PWM 计数器，分别存储在 RCAPDAT（上升沿锁存）和 FCAPDAT（下降沿锁存） 注意：只是针对 CH4	1'b0
[0]	RW	CAPINV2 捕获反向使能标识位 1'b0：捕获模式输入信号反向无效	1'b0

		1'b1：捕获模式输入信号反向有效，对输入信号取反 注意：只是针对 CH4	

24 QFLASH 控制器

24.1 功能概述

W800 内置 QFLASH 的控制器，提供总线方式的 QFLASH 读写擦操作，提供系统总线和数据总线的访问仲裁，实现 CACHE 方式的 QFLASH 读操作。

24.2 主要特性

- 提供总线访问 FLASH 接口
- 支持通过寄存器配置的方式对 FLASH 直接发送 SPI 命令进行访问
- 支持对 FLASH 进行 24 位或 32 位地址访问
- 支持对存储在 FLASH 中的代码和数据自动解密后读取

24.3 功能描述

24.3.1 总线访问

FLASH 在系统中地址从 0x08000000 开始，可以直接对该地址空间进行读操作。

24.3.2 寄存器访问

通过对寄存器的配置可以直接向 FLASH 发送 SPI 命令，以实现对 FLASH 更灵活的访问，支持大部分 FLASH 的控制命令。

24.3.3 命令的配置和启动

Flash 的控制命令码写入寄存器 FLASH_CMD 的 command 位，同时根据命令格式配置该寄存器的其它位，例如，如果命令中有地址位，要将 address 位置 1。如果命令中有数据，要将 DAT 位置 1，具

体参考寄存器说明。

命令配置完成后，将 CMD_START 寄存器的 command_b 位置 1，则控制器开始向 FLASH 发送该命令。

24.3.3.1 通信模式的及配置

首先向 FLASH 发送命令将 FLASH 配置为 QIO 或 QPI 模式，然后将 FLASH 控制器的 QIOM 或 QPIM 位置为 1，即可使用 QIO 模式或 QPI 模式与 FLASH 进行通信。

24.3.3.2 数据缓存

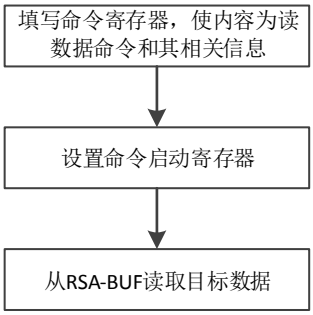
通过寄存器对 FLASH 进行数据读写时，使用 0x4000 0000 ~ 0x4000 03ff 地址的存储空间作为数据缓存，该段地址空间同时也作为 RSA 加解密模块使用的数据缓存。即读数据时，控制器将从 FLASH 读出的数据顺序存储在从 0x4000 0000 开始的地址空间进行缓存。向 FLASH 写数据时也是从 0x4000 0000 开始读取数据后发送给 flash。

24.3.3.3 读操作

每次读操作最多可读取 256 字节数据，可读取 QFlash 任意位置的数据。

读操作命令启动后，数据就存放在缓存中了，无需等待或查询。

操作流程如图所示。



24.3.3.4 写操作

以下命令均为写操作：

WRSR：写状态寄存器

PP：页编程

SE：sector 擦除

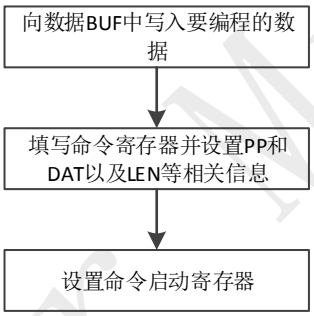
BE：32K block 擦除

HBE：64K block 擦除

CE：全芯片擦除

擦除命令不需要数据，直接将命令寄存器配置为相应命令，然后启动就可以了。

编程命令过程如下



24.4 寄存器描述

24.4.1 寄存器列表

表 201 QFLASH 控制器寄存器列表

偏移地址	名称	缩写	描述	默认值
0X0000_0000	命令信息寄存器	CMD_INFO	QFLASH 操作所需命令及数据字节	0X0000_0000
0X0000_0004	命令启动寄存器	CMD_START	命令启动	0X0000_0000
0X0000_0008	Flash 控制寄存器	Flash_CR	QFLASH 操作模式控制	0X0000_0000
0X0000_000C	Remap 寄存器	Remap	Remap 选项控制	0X0000_0000
0X0000_0010	ADDR 寄存器	Flash_ADDR	操作地址	0X0000_0000
0X0000_0014	解密控制寄存器	DECRYPT_CR	固件机密模式控制	0X0000_0000
0X0000_0018	解密状态寄存器	DECRYPT_STA	固件解密状态；	0X0000_0000
0X0000_0200 … 0X0000_0600	读写数据缓存区	RSA_BUF	用于缓存 QFLASH 读写的数据或者 RSA 的加解密数据。**此内存为 QFLASH 控制器和 RSA 模块共用，因此，QFLASH 读写与 RSA 操作不能同时使用，需要在驱动实现时增加互斥操作保护**	0X0000_0000

24.4.2 命令信息寄存器

表 202 QFLASH 命令信息存器

位	访问	操作说明	复位值

[31]	RW	1：指示 CMD_START 携带 Address，CMD_START 寄存器中存有该 Address 的低 20 位，该 Address 的高 4 为固定为 0，共计 24 位	1'b0
[30]	RW	1：指示 CMD_START 寄存器中 CRM 携带有内容	1'b0
[29]	RW	1：指示 CMD_INFO 携带 Dummy 周期	1'b0
[28:26]	RW	其内容+1 后指示 CMD_INFO 携带多少 Dummy 周期	3'b0
[25:16]	RW	其内容+1 后指示 CMD_INFO 携带多少字节的数据	10'b0
[15]	RW	1：表示 CMD_INFO 携带数据	1'b0
[14]	RW	1：表示当前 CMD_INFO 命令域填写的命令为读命令，命令包含 RDSR、FR、QIOFR、RDPDRID、RSR 等	1'b0
[13]	RW	1：表示当前 CMD_INFO 命令域填写的命令为 WRSR 命令	1'b0
[12]	RW	1：表示当前 CMD_INFO 命令域填写的命令为 PP 命令	1'b0
[11]	RW	1：表示当前 CMD_INFO 命令域填写的命令为 SE 命令	1'b0
[10]	RW	1：表示当前 CMD_INFO 命令域填写的命令为 BE 命令	1'b0
[9]	RW	1：表示当前 CMD_INFO 命令域填写的命令为 HBE 命令	1'b0
[8]	RW	1：表示当前 CMD_INFO 命令域填写的命令为 CE 命令	1'b0
[7 : 0]	RW	CMD_INFO 命令域，填写 QFlash 的命令，可参考 QFlash 芯片手册。	8'b0

24.4.3 命令启动寄存器

表 203 QFLASH 命令启动寄存器

位	访问	操作说明	复位值
[31:29]	RO	Reserved	3'b0

[28]	RW	置 1 表示启动命令，命令操作完成后，由硬件清零。	1'b0
[27: 8]	RW	Address[19:0]：若命令寄存器中的 A 标记为为 1，则此处存储 Address 低 20 位内容。	20'b0
[7 : 0]	RW	CRM[7:0]：若命令寄存器中的 CRM 标记为为 1，则此处存储 CRM 内容。	8'b0

24.5 QFLASH 的常用指令

表 204 QFALSH 常用命令

命令名称	命令字	命令缩写
Write Enable	06H	WREN
Write Disable	04H	WRDI
Read Status	05H	RDSR
Write Status	01H	WRSR
Fast Read	0BH	FR
Quad IO Fast Read	EBH	QIOFR
Page Program	02H	PP
Sector Erase	20H	SE

其它命令参见 QFLASH 的相关手册。

25 PSRAM 接口控制器

25.1 功能概述

W800 内置 SPI/QSPI 接口的 PSRAM 控制器，支持外置 PSRAM 设备访问，提供总线方式的 PSRAM 读写擦操作。最高读写速度 80MHz。

25.2 主要特性

- 支持对外置 PSRAM 的读写访问
- 可配置为 SPI 和 QSPI
- SPI/QSPI 时钟频率可配置
- 支持 BURST INC 模式访问
- 支持 PSRAM 的半休眠模式

25.3 功能描述

PSRAM 全称为 Pseudo static random access memory，指的是伪静态随机存储器。与传统 SRAM 相比，具备封装小，容量大，成本低等优势，在物联网应用中主要用于数据缓存方向。接口多为 SPI，QSPI 等。接口引脚主要有时钟信号 SCK，片选信号 CS 及 4 根双向数据 IO。W800 提供的 PSRAM 控制器可以支持 SPI/QSPI 接口的 PSRAM 的总线方式访问，工作时钟最高速率 80MHz，最大容量支持 64Mb。

25.3.1 **引脚说明**

SCLK：SPI 接口时钟，SCLK 周期由 PSRAM_CTRL[7:4]设置，可设置的分频值最小为 3，默认为 AHB 时钟的 4 分频。

CS：SPI 接口片选信号

SIO0：SPI 模式数据输入，QSPI 模式 SD[0]

SIO1：SPI 模式数据输出，QSPI 模式 SD[1]

SIO2：QSPI 模式 SD[2]

SIO3：QSPI 模式 SD[3]

25.3.2 访问模式设置

PSRAM 控制器支持对外置 PSRAM 的 SPI 和四线的 QSPI 两种访问模式，默认为 SPI。通过设置 PSRAM_CTRL[1]配置 SPI 的模式。

PSRAM_CTRL[1]默认为 0，即 SPI 模式，此时完成一次写操作需要 64 个 SCLK 周期，完成一次读操作需要 72 个 SCLK 周期。

如果 PSRAM 工作在 SPI 模式下,此时向 PSRAM_CTRL[1]写入 1 时,控制器会向 PSRAM 发送命令 35H，当读取 PSRAM_CTRL[1]为 1 时，表示命令发送完成，PSRAM 进入 QPI 模式，此时完成一次写操作需要 16 个 SCLK 周期，完成一次读操作需要 22 个 SCLK 周期。

如果 PSRAM 工作在 QPI 模式下,此时向 PSRAM_CTRL[1]写入 0 时,控制器会向 PSRAM 发送命令 F5H，当读取 PSRAM_CTRL[1]为 0 时，表示命令发送完成，PSRAM 进入 SPI 模式。

设置 PSRAM 工作模式必须在初始化操作完成后，不能同时设置。

25.3.3 PSRAM 初始化

第一次使用前需要在 PSRAM 上电稳定后，向寄存器 PSRAM_CTRL[0]写入 1，启动 PSRAM 复位初始化操作，即向 PSRAM 发送 66H 和 99H 命令。默认使用 SPI 模式发送，即需要 8 个 SCLK + tCPH + 8 个 SCLK 的时间。初始化完成后，硬件自动将 PSRAM_CTRL[0]清零。

初始化操作会将 PSRAM 恢复成 SPI 模式。

推荐的初始化流程为：

(1) 将 PSRAM_CTRL[0]写入 1

(2) 等待，直到 PSRAM_CTRL[0]自动清零

(3) 通过软复位使 PSRAM 控制器复位

(4) 重新设置 PSRAM_CTRL 的其它需要参数

25.3.4 PSRAM 的访问方法

对 PSRAM 进行读写的方式与普通 SRAM 相同，即向相应的总线地址写入/读出数据。

25.3.5 BURST 功能

通过设置 PSRAM_CTRL[2]可以使控制器支持 AHB 总线发起的 BURST，即当 AHB 总线上 HBURST 为 1/3/5/7 时，表示 AHB 总线启动一次地址递增的连续读/写，此时为了提高 PSRAM 的访问速度，控制器在完成一个 word 的读/写后并不会将 CS 拉高，而是直接读/写下一个 word 的数据。

OVERTIMER 寄存器用于设置 CS 为低的最大时间，单位为 HCLK 的周期数，每开始一次 BURST 操作，内部计数器从 0 开始计数，当计数器值大于设定数值时，完成当前 word 的读/写后，控制器就自动停止 BURST 操作，直接将 CS 变为高电平，如果此时 AHB 总线上还有数据需要读/写则会当成单独的 WORD 进行。

PSRAM 控制器不支持 WRAP 形式的 BURST，如果访问 PSRAM 会用到 WRAP BURST，请将 PSRAM_CTRL[2]设为 0。

25.4 寄存器描述

25.4.1 寄存器列表

表 205 PSRAM 控制器寄存器列表

偏移地址	名称	缩写	描述	默认值
0X0000_0000	控制寄存器	PSRAM_CTRL	PSRAM 控制器设置	0X0000_0000
0X0000_0004	超时控制寄存器	OVERTIMER	CS 超时控制	0X0000_0000

25.4.2 命令信息寄存器

表 206 PSRAM 控制设置寄存器

位	访问	操作说明	复位值
[31:12]	RO	RSV	'b0
[11]	RW	HSM，Halfsleep mode 使能 1：使能 PSRAM 半休眠模式 0：清除半休眠模式	1'b0
[10:8]	RW	tCPH,CS 高电平最短时间设置，单位为 AHB 时钟周期数，必须大于 1，具体时间根据不同的 psram 手册中说明进行设置，不清楚可以不修改默认值	3'd6
[7:4]	RW	SPI 分频设置 只能配置为 2 以上数值，写入的数值就是分频的倍数	4'd4
[3]	RO	RSV	
[2]	RW	INC_EN BURST 功能使能 1：支持 AHB 总线上的 BURST 功能 0：不支持 BURST 功能	1'b0

[1]	RW	QUAD 写 1 使能 PSRAM 的 QPI 模式，写 0 使能 SPI 模式 读该标志位可知道当前 PSRAM 为哪种模式。	1'b0
[0]	RW	PSRAM 复位 写 1 启动对 PSRAM 的复位操作，复位完成自动清零。	1'b0

25.4.3 超时控制寄存器

表 207 CS 超时控制寄存器

位	访问	操作说明	复位值
[11:0]	RW	超时寄存器设置，设置 CS 为低的最大时间，用于 BURST 模式	12'd0

26 ADC

26.1 ADC 功能概述

基于 Sigma-Delta ADC 的采集模块，完成最多 4 路模拟信号的采集，采样率通过外部输入时钟控制，可采集输入电压，也可采集芯片温度，支持输入校准和温度补偿校准。

26.2 ADC 主要特性

- 支持最多 4 路数据采集
- 支持 DMA 模块用于数据缓冲；
- 支持中断交互模式；
- 支持采集数据与输入数据比较功能
- 最高采样频率 1KHz；
- 支持单端输入模式和差分输入模式；

26.3 ADC 功能描述

26.3.1 模块基本结构

ADC 模块为 PGA+SDADC 结构，输入信号经过 PGA 进行预放大后，输入到 SDADC 处理，SDADC 设计为 16bit ADC。需要注意的是，芯片内 ADC 采用 2.5V LDO 供电，为保护内部电路，输入信号必须要满足 Vinmax＜2.5V，其他限制条件，详见 26.3.8 节。

26.3.2 通道选择

寄存器 0x04[11:8] 提供通道选择功能。W800 的 ADC 提供 4 通道信号的采集功能。通过设置寄存器 0x04[11:8]可以选择 4 个通道中任一通道进行单端信号数据采集，或者选择 4 通道中预先配对的 2 对差分通道进行差分信号数据采集。具体通道选择可以参见寄存器描述。

除了 4 个通道以外，ADC 模块还提供了温度检测，电压检测以及 offset 校准功能。分别对应通道选择的 4’b1100，4’b1101，4’b1110.

如果需要使用对应功能，将通道选择配置为对应的通道，然后按照以上使用说明的流程操作即可。

26.3.3 数据比较

ADC 模块提供数据比较功能，可通过配置寄存器 0x10 的 BIIT[5] 设置此功能的开关。

用户可通过结果寄存器 0x18 的 BIT[17:0] 配置希望比较的数值。此数值需要转换为 18bit 有符号数，最高位为符号位。ADC 配置寄存器 0x10 的 BIT[6] 可以设置数据比较的方向，当此位为 0 时，如果 ADC 采集的值大于或者等于 Comp_data，则 INT_CMP 会被置 1，并且会产生一个中断；当此位为 1 时，只有当 ADC 采集的值的小于 Comp_data，则 INT_CMP 会被置 1，并且产生一个中断。

26.3.4 PGA 增益调整说明

寄存器地址 0X08 的 BIT[8:4]，5 位增益控制信号，其中

0X08[8:7]=GAIN_CTRL_PGA<4:3>（对应 GAIN2）

0X08[6:4]=GAIN_CTRL_PGA<2:0>（对应 GAIN1，110 和 111 不使用）

PGA 整体增益 GAIN_PGA=GAN1*GAIN2

PGA 增益配置表（用放大倍数表示）

[8:7] / [6:4]	00	01	10	11
000	1	2	3	4
001	16	32	48	64
010	32	64	96	128
011	64	128	192	256
100	128	256	384	512
101	256	512	768	1024

根据仿真结果，同样的放大倍数，建议优先采用其中 GAIN2 放大倍数较大的配置。

26.3.5 单端模式 VCMIN 调整说明

PGA 内部 VCMIN 电压，在单端模式下，作为运放输入负端。

调整 VCMIN 的目的是适应各种不同偏置点的信号源，防止在某些过高或者过低的偏置点下 PGA 输出饱和。

寄存器地址 0X40000D20 的 BIT[16:11]，默认 100000，即 VCMIN default=1.311V， step≈39.1mV

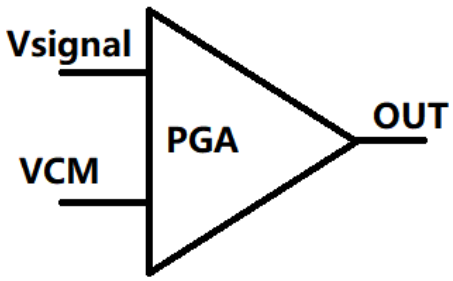


图 37 单端工作模式原理

26.3.6 Bypass 模式

Bypass_pga 寄存器地址 0X08 的 BIT[3]，默认 0，不旁路，PGA 正常工作；若为 1，此时 PGA 被旁路，外部信号源直接连接到 SDADC 输入端，主要用于内部模块测试，正常使用时不需要 bypass。

Bypass_ref 寄存器地址 0X08 的 BIT[2]，默认 0，不旁路，SDADC 正常工作；若为 1，此时 SDADC 内部 ref 被旁路，需要通过测试脚添加外部基准电压，主要用于内部模块测试，正常使用时不需要 bypass。

26.3.7 SDADC 输出码补充说明

SDADC 输出结果保存在 ADC_RESULT 寄存器中，寄存器地址 0X00 BIT[17:0]。

SDADC 本身输出码为无符号数，由于数字对 SDADC 的输出码进行了处理（最高位取反），变为有符号数，而 SDADC 输出码的 0 码并不固定（且不是正好为 10 0000 0000 0000 0000），所以直接采用寄存器的数值进行后续计算，是错误的。

对 SDADC 的输出结果进行计算时，寄存器 ADC_RESULT[17:0]的值（读取值是 16 进制数），需要软件先转换回无符号数（2 进制数最高位 ADC_RESULT[17]取反），再进行后续计算。

此次设计的 SDADC 为 16bit ADC（设计有效位数 16bit），而寄存器数据为 18bit，通过数字滤波器增加了最低两位，16bit 数转为 18bit 数(18bit 数≈16bit 数*4)，所以有效位数为高 16bit（即 ADC_RESULT[17:2]）。

在将读取值转回无符号数之后，涉及 LSB 的计算，还需要软件将 18bit 数据转为 16bit 数据，即舍去最低两位数据（ADC_RESULT[1:0]），只保留高 16bit（转回无符号数的 ADC_RESULT[17:2]），然后用 16bit 数据进行后续计算，此时 LSB 的仿真结果约等于 68.5uV（还需测试确定芯片实际 LSB）。

26.3.8 输入信号电压范围

因为 NTO 版本 PGA 和 SDADC 采用 2.5V LDO 供电，为防止内部电路饱和，对输入信号电压范围有限制。

假定输入差分信号为 Vinp 和 Vinn（单端模式时，信号 Vinn 为 VCMIN），则输入差分电压 Vdiff=Vinp-Vinn，输入共模电压 Vcm=(Vinp+Vinn)/2。PGA 第一级增益 GAIN1，第二级增益 GAIN2，PGA 整体增益 GAIN_PGA=GAN1*GAIN2（详见 2.2.2 节）。

输入信号需同时满足下列限制条件：

0V<Vinp<2.5V，0V<Vinn<2.5V；

0V<Vcm-(Vdiff*GAIN1)/2；

Vcm+(Vdiff*GAIN1)/2<2.5V；

0V<1.2V-(Vdiff*GAIN_PGA)/2；

1.2V+(Vdiff*GAIN_PGA)/2<2.5V；

对于已知输入信号大致范围的，可以直接代入上面的式子计算，选取可用增益设置。

对于未知输入信号，可先用 x1 增益设置，判断出大致范围，再代入上面的式子计算，选取可用增益设置。

26.4 典型使用情况的寄存器配置

以下配置字都为 16 进制数，高位的 0 已省略。

26.4.1 Offset 测量

1. 设置模块工作状态,将寄存器 0X04 设置为 0xE03，打开信道选择（offset 检测）/SDADC_CHOP 使能/SDADC 使能/LDO 使能

2. 设置模块寄存器 0X08=0x3，PGA 为 x1 增益配置/PGA_CHOP 使能/PGA 使能。其中，增益配置 0X08 BIT[8:4]要根据所进行测试的要求，进行相应的修改（即后面测试用的什么增益配置，offset 测量时也要采用相同的增益配置）

3. 设置时钟状态 0X40000E14 BIT[15:8]=0x28，确认 SDADC 时钟配置为 1MHz。芯片上电后默认值即为 0x28，若无改动，此步骤可省略

4. 等待 2ms，读出 ADC_RESULT 寄存器的输出结果，参照 26.3.7 节，需要软件先转换回无符号数（最高位 ADC_RESULT[17]取反）,再舍去最低两位,只保留高 16bit 数据。重复读取共 10 次,读数间隔 2ms，算出码数平均值记为 code_offset

26.4.2 差分输入模式

1. 先按照 26.4.1 节进行 offset 测量，得出 code_offset

2. 设置模块寄存器 0X04=0x803，信道选择（正 channel 0，负 channel 1）/SDADC_CHOP 使能/SDADC 使能/LDO 使能。其中信道选择 0X04[11:8]，可配置成其他 channel 的差分输入，参考 26.3.2 节;

3. 设置模块寄存器 0X08=0x3，PGA 为 x1 增益配置/PGA_CHOP 使能/PGA 使能。其中，增益配置 0X08[8:4]根据输入信号电压范围，选取合适的配置，参考 26.3.4 和 26.3.8 节

4. 设置时钟状态 0X40000E14[15:8]=0x28，确认 SDADC 时钟配置为 1MHz。芯片上电后默认值即为 28，若无改动，此步骤可省略

5. 等待 2ms，读出 ADC_RESULT 寄存器的输出结果，寄存器地址 0X00[17:0]。参照 26.3.7 节，需要软件先转换回无符号数（最高位 ADC_RESULT[17]取反），再舍去最低两位，只保留高 16bit 数据。6. 重复读取共 10 次，读数间隔 2ms，算出码数平均值记为 code_out

7. 差分输入信号 Vindiff=(code_out-code_offset)*LSB/GAIN_PGA，其中 LSB 根据仿真结果约等于 68.5uV

26.4.3 单端输入模式

1. 先按照 26.4.1 节进行 offset 测量，得出 code_offset;

2. 设置模块寄存器 0X04=0x3，信道选择（正 channel 0，负 VCMIN）/SDADC_CHOP 使能/SDADC 使能/LDO 使能。其中信道选择 0X04[11:8]，可配置成其他 channel 的单端输入，参考 26.3.2 节

设置模块寄存器 0X408=0x3，PGA 为 x1 增益配置/PGA_CHOP 使能/PGA 使能。其中，VCMIN 配置 0X40000D20[16:11]和增益配置 0X08[8:4]，根据输入信号电压范围，选取合适的配置，参考 26.3.4、26.3.5 和 26.3.8 节

3. 设置时钟状态 0X40000E14[15:8]=0x28，确认 SDADC 时钟配置为 1MHz。芯片上电后默认值即为 28，若无改动，此步骤可省略;

4. 等待 2ms，读出 ADC_RESULT 寄存器的输出结果，寄存器地址 0X00[17:0]。参照 26.3.7 节，需要软件先转换回无符号数（最高位 ADC_RESULT[17]取反），再舍去最低两位，只保留高 16bit 数据。重复读取共 10 次，读数间隔 2ms，算出码数平均值记为 code_out;

5. 单端模式输入信号 vin_single=(code_out-code_offset)*LSB/ GAIN_PGA+VCMIN，其中 LSB 根据仿真结果约等于 68.5uV，VCMIN 默认值 1.311V;

26.4.4 温度检测模式

1. 设置模块寄存器 0X04=0xC03，信道选择（温度检测）/SDADC_CHOP 使能/SDADC 使能/LDO 使能;

2. 设置模块寄存器 0X08=0x183，PGA 为 x4 增益配置/PGA_CHOP 使能/PGA 使能;

3. 设置时钟状态 0X40000E14[15:8]=0x28，确认 SDADC 时钟配置为 1MHz。芯片上电后默认值即为 28，若无改动，此步骤可省略；

4. 设置 tempsensor 模块工作状态 0X0C=0x1，TempSensor 使能/cal_offset_temp12=0，TempSensor 为 x2 增益配置；

5. 等待 2ms，读取 ADC_RESULT 寄存器的输出结果，参照 26.3.7 节，需要软件先转换回无符号数（最高位 ADC_RESULT[17]取反），记为输出码 code_out1;

6. 设置 tempsensor 模块工作状态 0X0C=0x3，TempSensor 使能/cal_offset_temp12=1，TempSensor 为 x2 增益配置；

7. 等待 2ms，读取 ADC_RESULT 寄存器的输出结果，参照 26.3.7 节，需要软件先转换回无符号数（最高位 ADC_RESULT[17]取反），记为输出码 code_out2；

8. 重复步骤 4 到步骤 7，共 10 次，分别算出 code_out1 和 code_out2 的平均值 code_out1avg 和 code_out2avg；

9. 由上面步骤算出 Vtemp=(code_out1avg-code_out2avg)/16，而 TempSensor 输出电压=Vtemp 码数*（LSB/4），其中 LSB 根据仿真结果约等于 68.5uV（此 LSB 是按 16bit 的仿真结果，温度检测上面的步骤 5 和 7 里，读取都是 18bit 数据，18bit 数≈16bit 数*4，则 18bit 对应的最低有效位≈LSB/4）

10. 根据测试结果，目前整理出了拟合公式，可直接估算芯片温度

Vtemp 码=(15.548*芯片温度)+4444.1

将步骤 9 算出的 Vtemp 代入上面的公式，得出芯片温度。（注意，拟合公式采用的是十进制）

另外，补充一个提高精度的可选措施：

对不同芯片，假定拟合公式 y=kx+b 中的系数 k 不变（都为 15.548），不同芯片的系数 b 有一定差别。可按照上面的使用说明，先在已知环境温度下（例如室温 25℃），算出 Vtemp 码数，代入 Vtemp 码=(15.548*(已知环境温度+11.8))+b，得出不同芯片的系数 b，保存在 rom 或 flash 中，替换上面公式的 4444.1。然后，按照新公式的系数计算温度，可在一定程度上提高不同芯片的温度检测精度。

26.4.5 电压检测模式

1. 先按照 26.4.1 节进行 offset 测量，得出 code_offset;

2. 设置模块寄存器 0X04=0xD03，信道选择（电压检测）/SDADC_CHOP 使能/SDADC 使能/LDO 使能

3. 设置模块寄存器 0X08=0x83，PGA 为 x2 增益配置/PGA_CHOP 使能/PGA 使能。

4. 设置时钟状态 0X40000E14[15:8]=28,确认 SDADC 时钟配置为 1MHz。芯片上电后默认值即为 28，若无改动，此步骤可省略

5. 等待 2ms，读出 ADC_RESULT 寄存器的输出结果，参照 26.3.7 节，需要软件先转换回无符号数（最高位 ADC_RESULT[17]取反),再舍去最低两位,只保留高 16bit 数据。重复读取共 10 次,读数间隔 2ms，算出码数平均值记为 code_out;

6. 芯片电源电压 Vpower=(code_out-code_offset)*LSB/2+1.2V，其中 LSB 根据仿真结果约等于 68.5uV

26.5 ADC 寄存器描述

26.5.1 寄存器列表

表 208 ADC 寄存器列表

偏移地址	名称	缩写	描述	复位值
0X0000	ADC 结果寄存器	ADC_RESULT	存放 ADC 采集值及数据比较值	0X0000_0000
0X0004	ADC 模拟寄存器	ADC_ANA_CTRL	配置 ADC 相关功能	0X0000_0004
0X0008	PGA 配置寄存器	PGA_CTRL	配置 PGA 相关功能；	0x0000_0000
0X000c	TempSensor 配置寄存器	TEMP_CTRL	配置温度传感器相关功能；	0x0000_0000
0x0010	ADC 配置寄存器	ADC_CTRL	配置 ADC 模块功能；	0x0050_0500
0x0014	ADC 中断寄存器	ADC_INT_STATUS	ADC 模块中断状态寄存器；	0x0000_0000
0x0018	比较值寄存器	CMP_VALUE	比较阈值设置	0x0000_0000

26.5.2 ADC 结果寄存器

表 209 ADC 结果寄存器

位	访问	操作说明	复位值
[17：0]	RW	ADC 转换结果值，有效 bit 位宽 18bits。有符号数，最高位为符号位。	1’b0

26.5.3 ADC 模拟配置寄存器

表 210 ADC 配置寄存器

位	访问	操作说明	复位值
[11 : 8]	RW	ADC 工作通道选择信号： 4’b0000：AIN0 通道工作。DMA 模式下，对应 DMA 通道 0 工作 4’b0001：AIN1 通道工作。DMA 模式下，对应 DMA 通道 1 工作 4’b0010：AIN2 通道工作。DMA 模式下，对应 DMA 通道 2 工作 4’b0011：AIN3 通道工作。DMA 模式下，对应 DMA 通道 3 工作 4’b0100：RSV 4’b0101：RSV 4’b0110：RSV 4’b0111：RSV 4’b1000：AIN0/AIN1 差分信号输入。DMA 模式下，对应 DMA 通道 0 工作 4’b1001：AIN2/AIN3 差分信号输入。DMA 模式下，对应 DMA 通道 2 工作 4’b1010：RSV 4’b1011：RSV 4’b1100：温度传感器输入，对应 DMA 通道 2 4’b1101：电压检测模块输入，对应 DMA 通道 3 4’b1110：offset 检测输入； 4’b1111：RSV	4’b1000
[7]	RO	RSV	1’b0
[6:5]	RW	chop_enr	2’b00

		LDO 斩波信号 PD 信号	
[4]	RW	Chop_ens sdadc 斩波信号 PD 信号 0：使能 1：不使能	1'b0
[3]	RO	RSV	1'b0
[2]	RW	Pd_sdadc sdadc 模拟模块掉电使能 1：掉电 0：工作	1'b1
[1]	RW	Rstn_sdadc 模拟模块数字逻辑部分复位信号 0：复位 1：正常工作	1'b0
[0]	RW	en_ldo_sdadc ADC LDO 使能 0：掉电 1：工作	1'b0

26.5.4 PGA 配置寄存器

表 211 PGA 配置寄存器

位	访问	操作说明	复位值
[8:4]	RW	Gain_ctrl_pga PGA 增益配置； BIT[8:7] 配置 GAIN2，BIT[6:4] 配置 GAIN1，具体增益表参考 Section26.3.4	5'd0
[3]	RW	Bypass_pga pga 旁路信号 1：旁路 pga 0：不旁路	1'b0
[2]	RW	Bypass_ref 内部参考电压旁路信号 1：旁路内部参考电压 0：不旁路	1'b0
[1]	RW	Chop_enp PGA 斩波使能信号 1：使能 0：不使能	1'b0
[0]	RW	En_pga Pga 使能信号 1：使能 0：不使能	1'b0

26.5.5 TEMP 配置寄存器

表 212 温度传感器配置寄存器

位	访问	操作说明	复位值
[5:4]	RW	Gain_temp temp 增益控制 编码 Gain 00 -- 2 01 -- 4 10 -- 6 11 -- 8	2'd0
[3：2]	RO	RSV	2'b0
[1]	RW	Cal_offset_temp12 TEMPSEN offset 校准功能	1'b0
[0]	RW	ON_TEMP 1：temp 使能 0：temp 不使能	1'b0

26.5.6 ADC 功能配置寄存器

表 213 温度传感器配置寄存器

位	访问	操作说明	复位值
[29:20]	RW	ana_swth_time 软件切换数据通道之后，模拟电路保持稳定需要的时间，默认为 80 个 pclk，即 2us	10'h050

[19:18]	RO	RSV	2'b0
[17: 8]	RW	ana_init_time 软件启动 adc_start 后，模拟电路保持稳定需要的时间，默认为 80 个 pclk，即 2us	10'h050
[7]	RO	RSV	1'b0
[6]	RW	Cmp_pol 0: adc_result >=cmp_value 时产生中断 1：adc_result<cmp_value 时产生中断	1'b0
[5]	RW	Cmp_int_ena 1：比较中断使能 0：比较中断不使能	1'b0
[4]	RW	Adc_cmp_enable 1：adc 比较功能使能 0：adc 比较功能不使能	1'b0
[3:2]	RO	Reserved	2'b0
[1]	RW	Adc_int_ena 1：ADC 数据转换中断使能 0：ADC 数据转换中断不使能	1'b0
[0]	RW	Adc_dma_enable 1：dma 使能 0：dma 不使能	1'b0

26.5.7 ADC 中断状态寄存器

表 214 ADC 中断状态寄存器

位	访问	操作说明	复位值
[1]	RW	Cmp_int 比较中断标志位，硬件置位，软件写 1 清零	1'b0
[0]	RW	Adc_int 数据转换完成中断，硬件置位，软件写 1 清零	1'b0

26.5.8 比较阈值寄存器

表 215 比较阈值寄存器

位	访问	操作说明	复位值
[17:0]	R/W	Cmp_value 要比较的数值	18'h00000

27 Touch Sensor

27.1 模块功能概述

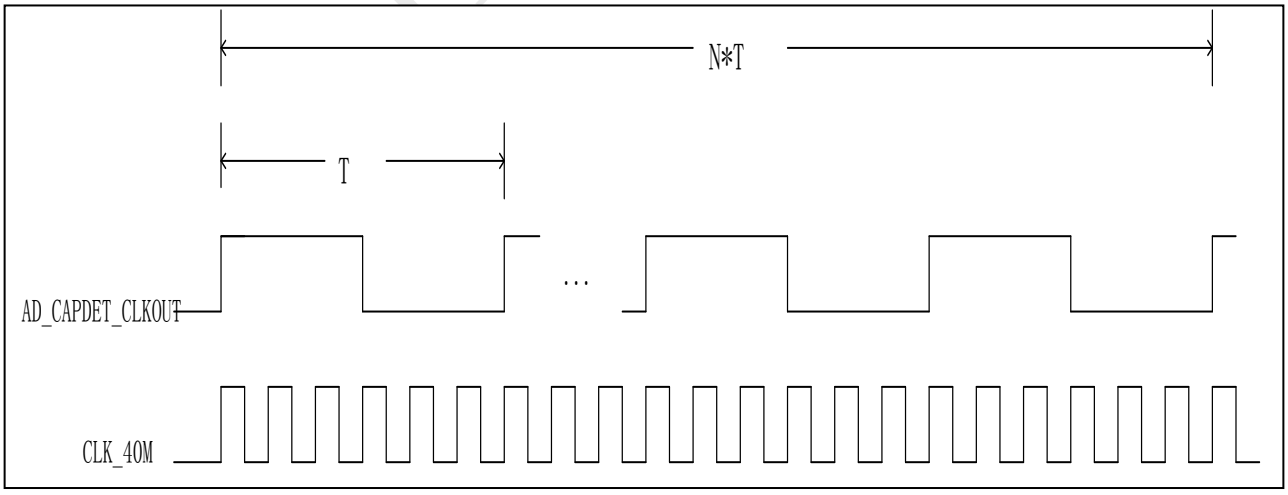
模块基本功能如下：

- 支持最多 16 路 Touch Sensor 扫描；
- 记录每路 Touch Sensor 扫描结果；
- 通过中断上报扫描结果；

27.2功能使用说明

W800 系统中集成触摸按键模块，基本原理是当按键被触摸时，按键上电路的电容值会发生变化，从而影响模块中输出时钟频率。通过检测模块的输出时钟的频率，可以判断电容值是否发生变化，从而判断按键的状态。

此数字模块的基本功能为每隔一定时间逐次扫描每个触摸按键的状态，在设定的时间窗口内计数并记录下每个按键的状态。如果超过设定的阈值，则判断该按键被触摸，通过中断上报给 MCU 系统。



27.2.1 基本工作流程

1. 设置 Touch_CR 寄存器，配置扫描周期，扫描窗口，以及选择需要扫描的 IO。Touch_CR 中 scan_period 为每次扫描的间隔周期，单位为 16ms。即如果寄存器设置为 10，则每 160ms 将会逐次扫描 16 个触摸按键。CAPDET_CNT 为扫描每个 IO 状态时计数的窗口，即上图的 N。需要注意的是，为避免切换通道导致的抖动，每次切换扫描 IO 后在第三个脉冲开始才会开始计数，因此如过该寄存器设置为 N，则实际计数窗口为 N-2。

2. 设置每个 Touch IO 对应的计数阈值。如果扫描的结果超过基数+阈值，则可认为该按键被触摸；

3. 使能 Touch_CR[0]，模块开始工作。

4. 在使能触摸按键模块后，硬件会先逐次扫描选中的 IO，并将各个 IO 的计数值存储下来，作为基数。然后每隔一个 scan_period 逐次扫描选中的 IO，将当前扫描的值存储下来。所有 IO 扫描完毕以后会将每个 IO 的计数值与基数做比较，如果当前值>基数+阈值，则认为此按键被触摸，寄存器 0x44 中 PAD_STATUS 中对应位会被置位为 1，并通过中断上报给系统。PAD_STAUS 写 1 清 0.

27.3 寄存器列表：

表 216 Touch Sensor 控制器寄存器列表

偏移地址	名称	缩写	描述	默认值
0X0000_0000	控制寄存器	Touch_CR	Touch Sensor 控制器设置	0X0000_0000

0X0000_0004 ~ 0X0000_0040	触控按键单路控制	Touch_Sensor x	每路触控按键阈值控制与计数值	0X0000_0000
0x0000_0044	中断控制器	Int_Source	中断控制器	0x0000_0000

27.3.1 Touch Sensor 控制寄存器

表 217 Touch Sensor 控制设置寄存器

位	访问	操作说明	复位值
[31:26]	RW	Scan_period 扫描周期，单位为 16ms；例如 scan_period 设置为 6'd10，则每 160ms 扫描一次按键状态；	6'd10
[25:20]	RW	CAPDET_CNT 选择 CAPDET 输出脉冲数作为计数窗口。 注： 为避免切换通道导致的抖动，在第三个脉冲开始才会开始计数，因此如此寄存器设置为 N，则实际计数窗口为 N-2。例如设置为 6'd20 ，则以 18 个 CAPDET 脉冲的周期为计数窗口。	6'd20
[19:4]	RW	Touch Sensor 按键选择；扫描对应 bit 的触摸按键状态。 0x0000：不扫描； 0x0001：扫描第一个按键； 0x0002：扫描第二个按键；	16'd0

		0x0003：扫描第一个和第二个按键； … 0xFFFF：扫描所有 16 个 按键；	
[3:1]	RW	RSV	3’d0
[0]	RW	触摸按键控制器使能 1： 使能触摸按键扫描； 0： 关闭触摸按键扫描；	1’b0

27.3.2 触摸按键单路控制寄存器

表 218 触摸按键单路设置寄存器

位	访问	操作说明	复位值
[22:8]	RO	Touch Senor 1 计数值	14’d0
[7]	RO	RSV	1’b0
[6:0]	RW	Touch Senor 1 阈值	7’d50

27.3.3 中断控制寄存器

表 219 触摸按键中断控制寄存器

位	访问	操作说明	复位值
[31:16]	RW	INT_EN 对应 bit 为 1 时表示对应 IO 被触发会产生中断； 对应 bit 为 1 时表示对应 IO 被触发不会产生中断；	16’d0

[15:0]	RW	PAD_STATUS/INT_SOURCE bit 为 1 时表示对应 PAD 被触发； bit 为 0 时表示对应 PAD 未触发； 写 1 清 0	16'd0

28 W800 安全架构设计

28.1 功能概述

XT804 通过 HPROT 信号表示总线访问的安全特性

	0	1
HPROT[3]	uncacheable	cacheable
HPROT[2]	un-security	security
HPROT[1]	User	super
HPROT[0]	Code	data

W800 对安全架构的支持分为对 sram 的安全访问控制和对外设的访问控制。

28.1.1 SRAM 安全访问控制器(SASC)

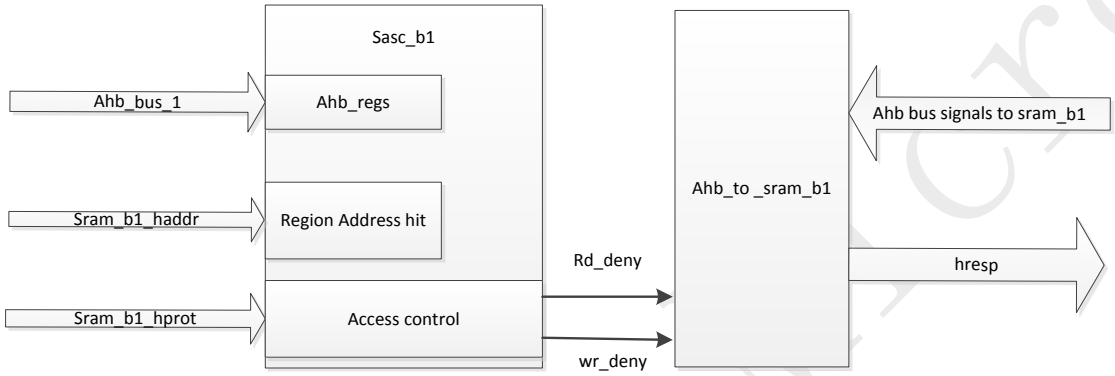
一级总线 SRAM，二级总线 SRAM 和 FLASH 各有一个安全访问控制器，每个 SASC 控制的存储空间都有如下的安全特性

- 8 个可配置的安全区域
- 每个安全区域最小可配置为 4 个字节
- Security-super 可以对所有区域进行没有限制的访问
- 如果 CPU 通过 AHB 总线上的访问被拒绝会产生错误的应答信号（HRESP=1），从而引发访问异常。
- 如果是总线上其它主设备比如 DMA 等访问被拒绝不会产生异常。

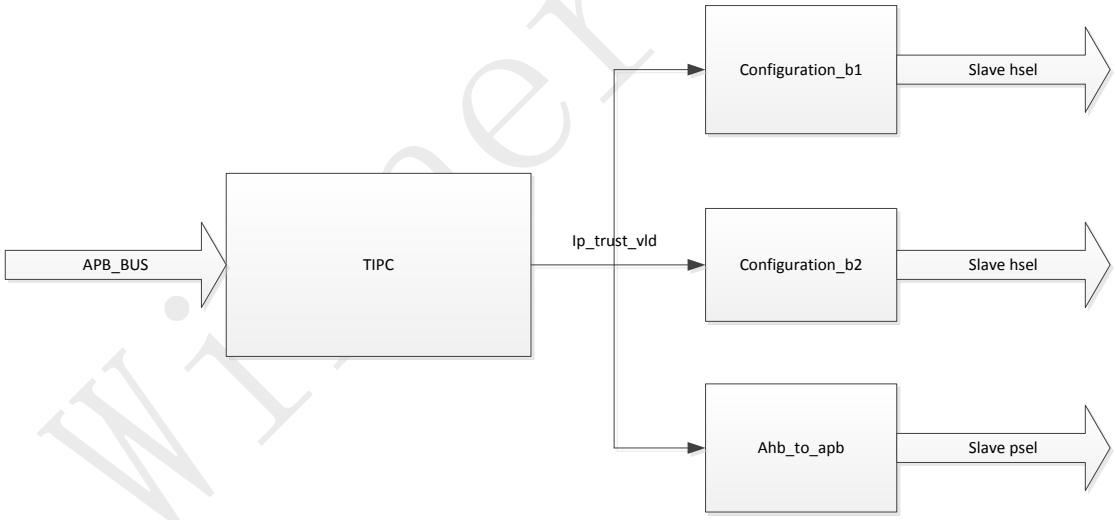
28.1.2 可信 IP 控制器（TIPC）

挂在 APB 总线上，用于配置包括一级总线，二级总线和 APB 总线上所有外设的信任权限。如果外设被配置为可信设备（ip_trust_vld=1），则只有总线上的访问是 security（HPROT[2]=1 或 PPROT[2]=1）时，才能正常访问外设，否则读写都会被拒绝。

28.2 安全架构框图



Sasc_b1功能框图



TIPC功能框图

28.3 寄存器说明

28.3.1 SASC 寄存器列表

名称	偏移地址	位阈	说明	复位值
CAR	0x0	[31:16]	保留未用	--
		[15:14]	sram region7 配置属性，同 region0	2'b00
		[13:12]	sram region6 配置属性，同 region0	2'b00
		[11:10]	sram region5 配置属性，同 region0	2'b00
		[9:8]	sram region4 配置属性，同 region0	2'b00
		[7:6]	sram region3 配置属性，同 region0	2'b00
		[5:4]	sram region2 配置属性，同 region0	2'b00
		[3:2]	sram region1 配置属性，同 region0	2'b00
		[1:0]	sram region0 配置属性 00: un-security-user 01: un-security-super 10: security-user 11: security-super	2'b00
CR	0x4	31:8	未用保留	--
		7	region7 属性控制寄存器，同 region0	
		6	region6 属性控制寄存器，同 region0	
		5	region5 属性控制寄存器，同 region0	
		4	region4 属性控制寄存器，同 region0	
		3	region3 属性控制寄存器，同 region0	

		2	region2 属性控制寄存器，同 region0	
		1	region1 属性控制寄存器，同 region0	
		0	region0 属性控制寄存器 0：属性无效　1：属性有效	
AP0	0x8	[31:16]	保留未用	
		[15:14]	region7 AP 配置，适用于 un-security-user	
		[13:12]	region6 AP 配置，适用于 un-security-user	
		[11:10]	region5 AP 配置，适用于 un-security-user	
		[9:8]	region4 AP 配置，适用于 un-security-user	
		[7:6]	region3 AP 配置，适用于 un-security-user	
		[5:4]	region2 AP 配置，适用于 un-security-user	
		[3:2]	region1 AP 配置，适用于 un-security-user	
		[1:0]	region0 AP 配置，适用于 un-security-user 00：R/W　01:RO　10:WO　11: No	

			Access	
CD0	0xC	[31:16]	保留未用	
		[15:14]	region7 CD 配置，适用于 un-security-user	
		[13:12]	region6 CD 配置，适用于 un-security-user	
		[11:10]	region5 CD 配置，适用于 un-security-user	
		[9:8]	region4 CD 配置，适用于 un-security-user	
		[7:6]	region3 CD 配置，适用于 un-security-user	
		[5:4]	region2 CD 配置，适用于 un-security-user	
		[3:2]	region1 CD 配置，适用于 un-security-user	
		[1:0]	region0 CD 配置，适用于 un-security-user 00: Date Access , Opcode Feche 01: Data Access 10: Opcode Feche 11: Data, Opcode all deny	

AP1	0x10	[31:16]	保留未用	
		[15:14]	region7 AP 配置，适用于 un-security-super	
		[13:12]	region6 AP 配置，适用于 un-security-super	
		[11:10]	region5 AP 配置，适用于 un-security-super	
		[9:8]	region4 AP 配置，适用于 un-security-super	
		[7:6]	region3 AP 配置，适用于 un-security-super	
		[5:4]	region2 AP 配置，适用于 un-security-super	
		[3:2]	region1 AP 配置，适用于 un-security-super	
		[1:0]	region0 AP 配置，适用于 un-security-super 00：R/W 01:RO 10:WO 11: No Access	
CD1	0x14	[31:16]	保留未用	
		[15:14]	region7 CD 配置，适用于 un-security-super	

		[13:12]	region6 CD 配置，适用于 un-security-super	
		[11:10]	region5 CD 配置，适用于 un-security-super	
		[9:8]	region4 CD 配置，适用于 un-security-super	
		[7:6]	region3 CD 配置，适用于 un-security-super	
		[5:4]	region2 CD 配置，适用于 un-security-super	
		[3:2]	region1 CD 配置，适用于 un-security-super	
		[1:0]	region0 CD 配置，适用于 un-security-super 00: Date Access , Opcode Feche 01: Data Access 10: Opcode Feche 11: Data, Opcode all deny	
AP2	0x18	[31:16]	保留未用	
		[15:14]	region7 AP 配置，适用于 security-user	
		[13:12]	region6 AP 配置，适用于 security-user	
		[11:10]	region5 AP 配置，适用于 security-user	

		[9:8]	region4 AP 配置，适用于 security-user	
		[7:6]	region3 AP 配置，适用于 security-user	
		[5:4]	region2 AP 配置，适用于 security-user	
		[3:2]	region1 AP 配置，适用于 security-user	
		[1:0]	region0 AP 配置，适用于 security-user 00：R/W 01:RO 10:WO 11: No Access	
CD2	0x1C	[31:16]	保留未用	
		[15:14]	region7 CD 配置，适用于 security-user	
		[13:12]	region6 CD 配置，适用于 security-user	
		[11:10]	region5 CD 配置，适用于 security-user	
		[9:8]	region4 CD 配置，适用于 security-user	
		[7:6]	region3 CD 配置，适用于 security-user	
		[5:4]	region2 CD 配置，适用于 security-user	
		[3:2]	region1 CD 配置，适用于 security-user	
		[1:0]	region0 CD 配置，适用于 security-user 00: Date Access , Opcode Feche 01: Data Access 10: Opcode Feche 11: Data, Opcode all deny	
REGION0	0x20	31:n+1	保留	0
		n:8	BAddr0，region0 基地址，n 与 sram 大	

			小相关， sram=32kB，n=20 sram=64kB, n=21	
		7:5	保留	3’b000
		4:0	RSize，region0 size 00101: 4B 00110: 8B …… 10001: 16KB 10010: 32KB ……	
REGION1	0x24	31:n+1	保留	0
		n:8	Baddr1，region1 基地址	
		7:5	保留	3’b000
		4:0	region1 size	
REGION2	0x28	31:n+1	保留	0
		n:8	region2 基地址	
		7:5	保留	3’b000
		4:0	region2 size	
REGION3	0x2C	31:n+1	保留	0
		n:8	region3 基地址	
		7:5	保留	3’b000

		4:0	region3 size	
REGION4	0x30	31:n+1	保留	0
		n:8	region4 基地址	
		7:5	保留	3'b000
		4:0	region4 size	
REGION5	0x34	31:n+1	保留	0
		n:8	region5 基地址	
		7:5	保留	3'b000
		4:0	region5 size	
REGION6	0x38	31:n+1	保留	0
		n:8	region6 基地址	
		7:5	保留	3'b000
		4:0	region6 size	
REGION7	0x3C	31:n+1	保留	0
		n:8	region7 基地址	
		7:5	保留	3'b000
		4:0	region7 size	

28.3.2 TIPC 寄存器

名称	偏移地址	位阈	说明	复位值
ip_trust_vld0	0x0	31:18	未用	0
		17	BT modem 可信属性 1：可信　0：不可信	0
		16	I2S 可信属性	0
		15	PWM 可信属性	0
		14	LCD driver 可信属性	0
		13	RF controller 可信属性	0
		12	timer 可信属性	0
		11	watch dog 可信属性	0
		10	PORTB 可信属性	0
		9	PORTA 可信属性	0
		8	UART5 可信属性	0
		7	UART4 可信属性	0
		6	UART3 可信属性	0
		5	UART2 可信属性	0
		4	UART1 可信属性	0
		3	UART0 可信属性	0

		2	SPI MASTER 可信属性	0
		1	SAR ADC 可信属性	0
		0	I2C 可信属性	0
ip_trust_vld1	0x4	31:18	未用	0
		17	RF BIST 可信属性配置 1：可信　0：不可信	0
		16	SDIO Wrapper 可信属性	0
		15	SPI_HS 可信属性	0
		14	SDIO 可信属性	0
		13	未用	0
		12	SEC 可信属性	0
		11	MAC 可信属性	0
		10	BBP 可信属性	0
		9	MMU 可信属性	0
		8	时钟复位控制模块可信属性	0
		7	PMU 可信属性	0
		6	BT 可信属性	0
		5	GPSEC 可信属性	0
		4	DMA 可信属性	0
		3	RSA 可信属性	0
		2	PSRAM 控制器可信属性	0
		1	FLASH 控制器可信属性	0

		0	SDIO HOST 可信属性	0

28.4 使用说明

28.4.1 内存安全访问（SASC）

28.4.1.1 寄存器的访问权限

SASC 寄存器的访问按照以下原则：

- CAR 寄存器只能在 cpu 为 security-super 时进行读写。
- 当 cpu 处于 un-security-super 时，可以访问被配置为 un-security-user 的 region 相关寄存器位。
- 当 cpu 为 security-super 时，可以读写所有寄存器。
- 其它情况下寄存器都不可访问

例如当 CAR 设为 16'he4e4 时，表示 region0 和 region4 被设为 un-security-user，此时如果 cpu 处于 un-security-super，则 cpu 可以读写寄存器 REGION0 和 REGION4，以及 CR[0], CR[4], APx[1:0], APx[9:8], CDx[1:0], CDx[9:8]。

28.4.1.2 保护区间地址的设置

每个 SASC 都支持 8 个可配置的存储器保护区间 region，REGIONx[31:8]中的基地址为想要配置的存储区间实际物理地址的 25 到 2 位，同时，Size 的大小和基地址需要满足下面的要求：

SIZE

00101: 4B;

00110: 8B;　　Baddrx[0] = 0

00111: 16B;　Baddrx[1:0] = 0

01000: 32B;　Baddrx[2:0] = 0

01001: 64B;　Baddrx[3:0] = 0

01010: 128B;　Baddrx[4:0] = 0

01011: 256B;　Baddrx[5:0] = 0

01100: 512B;　Baddrx[6:0] = 0

01101:1KB;　　Baddrx[7:0] = 0

01110: 2KB;　　Baddrx[8:0] = 0

01111:4KB;　　Baddrx[9:0] = 0

10000: 8KB;　　Baddrx[10:0] = 0

10001: 16KB;　Baddrx[11:0] = 0

例如，如果将 20000100 作为 region0 的基地址，那么 region0 最大可以设为 256B，如果想将 region0 设为 128B,REGION0 寄存器应填入 0x0000400a。如果将 20040000 作为基地址,则该 region 最大可设为 64KB，如果就要定义一个 64KB 的 region，则该寄存器应填入 0x01000013。

28.4.1.3 存储器的访问权限

Priority of the four authority for mem:

request mem	Security super	Security user	Un-Security super	Un-Security user
Security super	√+	√+	√+	√+
Security user	X	√	X	X
Un-Security super	X	X	√	√+
Un-Security user	X	X	X	√

Note: √+ all access, include read, write, data access, opcode fetch

√ access based on the current attribute

X no access

根据上图

- security-super 的总线访问可以随意访问 sram
- un-security-super 的总线访问可以随意访问被配置为 un-security-user 的 region
- security-user 的总线访问只能按当前 AP 和 CD 属性访问 security-user 的 region
- un-security-super 的总线访问可以按当前 AP 和 CD 属性访问 un-security-super 的 region
- un-security-user 的总线访问只能按当前 AP 和 CD 属性访问 un-security-user 的 region

APx 和 CDx 寄存器用于设置每个 region 对应于不同权限时的访问属性，其中 CDx 寄

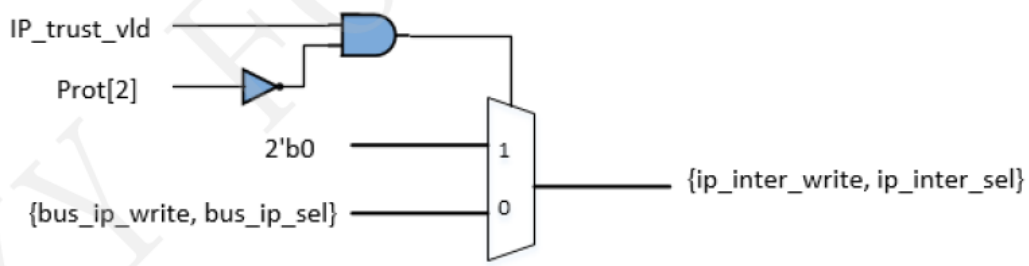
存器只在总线访问时读操作时有效，也就是设置是否允许读数据和读指令。APx 寄存器用于设置是否允许读/写操作。

例如如果 CAR[1:0]设为 00，并且 CR[0]为 1，表示 REGION 0 为 un-security-user，并且使能权限保护，此时如果总线访问为 security-super 或 un-security-super，则可以随意访问 REGION 0；如果总线访问为 Security-user，则任何访问都会被拒绝；如果总线访问为 un-security-user，AP0[1:0]为 01，表示只读，CD0[1:0]为 01，表示数据访问，则 REGION 0 允许总线读数据，其它操作均被拒绝。

如果 AHB 总线的访问了不能访问的区域或权限不对，则 sasc 模块会给出 read deny 或 write deny 信号，从而返回错误的 HRESP 应答信号，如果发起总线访问的是 CPU，则会产生硬件异常中断，如果是其它设备，则只会拒绝访问。

28.4.2 外设的可信访问

TIPC 模块只有 PROT[2]信号为 1 时才能写入，即只有在可信世界才能写入各个 IP 的可信属性配置寄存器。Ip_trust_vld 寄存器默认都为 0，即所有外设都是不可信的，此时外设可以随意访问，如果外设对应的 ip_trust_vld 被置位，也就是把外设设为可信设备，则只有可信世界的命令可以访问该外设。



29 附录 1. 芯片引脚定义

29.1芯片引脚分布

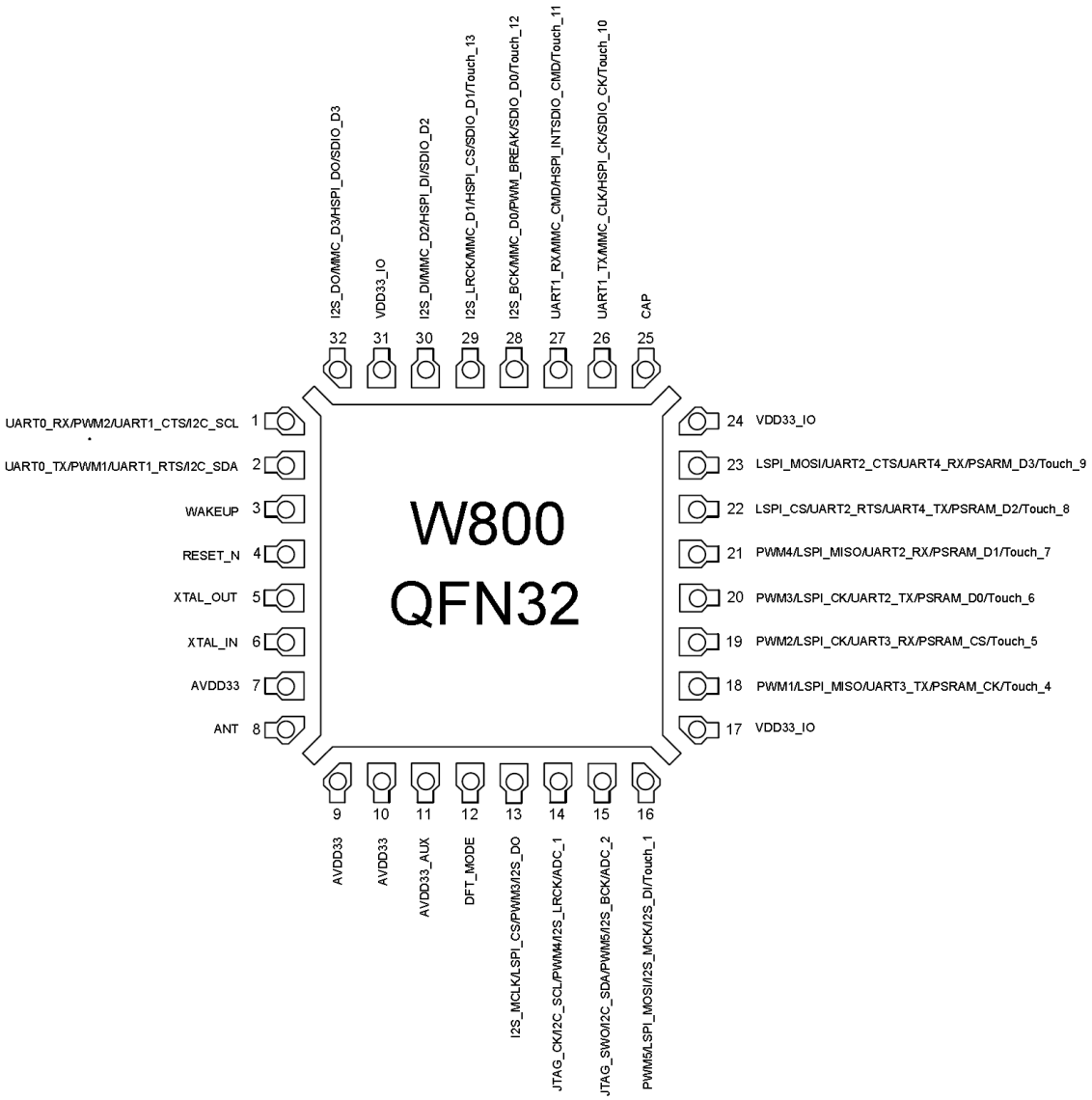


图 38 W800 芯片引脚分布

29.2芯片引脚复用关系

表 220 芯片引脚复用关系

编号	名称	类型	复位后管脚功能	复用功能	最高频率	上下拉能力	驱动能力
1	PB_20	I/O	UART_RX	UART0_RX/PWM2/UART1_CTS/I2C_SCL	10MHz	UP/DOWN	12mA
2	PB_19	I/O	UART_TX	UART0_TX/PWM1/UART1_RTS/I2C_SDA	10MHz	UP/DOWN	12mA
3	WAKEUP	I	WAKEUP 唤醒功能			DOWN	
4	RESET	I	RESET 复位			UP	
5	XTAL_OUT	O	外部晶振输出				
6	XTAL_IN	I	外部晶振输入				
7	AVDD33	P	芯片电源，3.3V				
8	ANT	I/O	射频天线				
9	AVDD33	P	芯片电源，3.3V				
10	AVDD33	P	芯片电源，3.3V				
11	AVDD33_AUX	P	芯片电源，3.3V				
12	TEST	I	测试功能配置管脚				
13	BOOTMODE	I/O	BOOTMODE	I2S_MCLK/LSPI_CS/PWM3/I2S_DO	20MHz	UP/DOWN	12mA
14	PA_1	I/O	JTAG_CK	JTAG_CK/I2C_SCL/PWM4/I2S_LRCK/ADC_1	20MHz	UP/DOWN	12mA
15	PA_4	I/O	JTAG_SWO	JTAG_SWO/I2C_SDA/PWM5/I2S_BCK/ADC_2	20MHz	UP/DOWN	12mA
16	PA_7	I/O	GPIO,输入，高阻	PWM5/LSPI_MOSI/I2S_MCK/I2S_DI/Touch_1	20MHz	UP/DOWN	12mA

17	VDD33IO	P	IO 电源，3.3V				
18	PB_0	I/O	GPIO,输入，高阻	PWM1/LSPI_MISO/UART3_TX/PSRAM_CK/Touch_4	80MHz	UP/DOWN	12mA
19	PB_1	I/O	GPIO,输入，高阻	PWM2/LSPI_CK/UART3_RX/PSRAM_CS/Touch_5	80MHz	UP/DOWN	12mA
20	PB_2	I/O	GPIO,输入，高阻	PWM3/LSPI_CK/UART2_TX/PSRAM_D0/Touch_6	80MHz	UP/DOWN	12mA
21	PB_3	I/O	GPIO,输入，高阻	PWM4/LSPI_MISO/UART2_RX/PSRAM_D1/Touch_7	80MHz	UP/DOWN	12mA
22	PB_4	I/O	GPIO,输入，高阻	LSPI_CS/UART2_RTS/UART4_TX/PSRAM_D2/Touch_8	80MHz	UP/DOWN	12mA
23	PB_5	I/O	GPIO,输入，高阻	LSPI_MOSI/UART2_CTS/UART4_RX/PSARM_D3/Touch_9	80MHz	UP/DOWN	12mA
24	VDD33IO	P	IO 电源，3.3V				
25	CAP	I	外接电容，1μF			-	
26	PB_6	I/O	GPIO，输入，高阻	UART1_TX/MMC_CLK/HSPI_CK/SDIO_CK/Touch_10	50MHz	UP/DOWN	12mA
27	PB_7	I/O	GPIO，输入，高阻	UART1_RX/MMC_CMD/HSPI_INT/SDIO_CMD/Touch_11	50MHz	UP/DOWN	12mA
28	PB_8	I/O	GPIO，输入，高阻	2S_BCK/MMC_D0/PWM_BREAK/SDIO_D0/Touch_12	50MHz	UP/DOWN	12mA
29	PB_9	I/O	GPIO，输入，高阻	I2S_LRCK/MMC_D1/HSPI_CS/SDIO_D1/Touch_13	50MHz	UP/DOWN	12mA
30	PB_10	I/O	GPIO，输入，高阻	I2S_DI/MMC_D2/HSPI_DI/SDIO_D2	50MHz	UP/DOWN	12mA
31	VDD33IO	P	IO 电源，3.3V				
32	PB_11	I/O	GPIO，输入，高阻	2S_DO/MMC_D3/HSPI_DO/SDIO_D3	50MHz	UP/DOWN	12mA
33	GND	P	接地				

声明

北京联盛德微电子有限责任公司保留随时修改、更新联盛德微电子产品或者各种文档、资料的权利。所有更新会通过联盛德微电子官方渠道进行发布。使用者必须确保通过官方渠道获取正确的信息。联盛德微电子不承诺将更新信息对所有人进行通知。